MACHINE PERCEPTION AND COMPUTER VISION

George I. Dimas

Department of Computer Science and Biomedical Informatics

University of Thessaly

A thesis submitted for the degree of

Doctor of Philosophy

December 2022

SUPERVISOR

Dimitris K. Iakovidis, *Professor*
University of Thessaly

ADVISORY COMMITTEE MEMBERS

Dimitris Iakovidis, *Professor*
University of Thessaly

Vassilis Plagianakos, *Professor*
University of Thessaly

Konstantinos Delibasis, *Associate Professor*
University of Thessaly

REVIEWING COMMITTEE MEMBERS

Dimitris Iakovidis, *Professor*
University of Thessaly

Vassilis Plagianakos, *Professor*
University of Thessaly

Konstantinos Delimpasis, *Associate Professor*
University of Thessaly

Michael Savelonas, *Assistant Professor*
University of Thessaly

Christos Anagnostopoulos, *Professor*
University of the Aegean

Ilias Maglogiannis, *Professor*
University of Piraeus

Ioannis Pratikakis, *Professor*
Democritus University of Thrace

ABSTRACT

Machine Learning (ML) is the basis of machine perception and computer vision. Machine perception refers to a machine's ability to comprehend various aspects of the world after processing and analysis of input data. Machine perception through visual data implements the concept of computer vision. The scientific contributions of this dissertation cover a wide range of methods and applications in this context. These include salient object detection and localization, image-based obstacle detection and avoidance, single-image visual measurements, enhancement of 3D point cloud object representations, and perceptually interpretable ML.

The prediction of visual attention on medical images is a research subject that has been limitedly studied. To tackle this problem, a novel and robust gaze estimation methodology based on physicians' eye fixations, using CNNs along with a novel co-operative training scheme is proposed, and a novel saliency dataset based on the eye fixations of physicians has been created. The approximation of the visual attention of humans has been a popular research topic; nevertheless, some of the best performing approaches require both RGB and sensor-based Depth (RGB-D) information. The need for additional sensors adds a complexity overhead to a system limiting its applicability. In this dissertation, the impact of accurate depth estimation for saliency perception is investigated, resulting in a novel monocular salient object detection (MonoSOD), based on a two-branch Convolutional Neural Network (CNN) autoencoder architecture, capable of predicting depth maps and estimating saliency through a trainable refinement scheme.

Another contribution of this dissertation is a novel methodology for obstacle detection based on RGB-D images. This methodology efficiently incorporates fuzzy logic and human eye fixations predicted using Generative Adversarial Networks (GANs). This combination can translate the position of detected obstacles into descriptive linguistic encodings that can be used in a variety of applications, such as robotic, and assistive navigation. An end-to-end self-supervised CNN model, with the capacity to simulate that RGB-D based obstacle detection method has been developed, requiring a standard RGB image as input.

The limitations of the current visual measurement methods, such as their requirements in terms of the number of input images and existence of reference objects, have motivated the development of a state-of-the-art, more robust method for single-image visual size measurements, named Virtual Grid Mapping (VGM). The proposed VGM method requires only a single image as input, and it does not require any prior information concerning the scene like the horizon line or reference objects. A major advantage of VGM that contributes to its robustness is that copes with the uncertainty originating from the calibration and the positioning of the camera; thus, offering a higher accuracy than current visual size measurement methods. Given the geometric properties of the camera, VGM automatically generates and projects a grid of virtual 3D points to the 2D image plane, enabling the establishment of approximative correspondences between 3D points of the real world and 2D points of the image plane. These correspondences enable the assessment of the distance between an object and a camera and subsequently the measurement of that object. A

similar approach to VGM assisted by the predicted depth provided by a CNN model is proposed for estimating the size of lesions in biomedical images of the gastrointestinal tract. Various studies confirmed that endoscopic assessment of lesion size has inherent limitations and significant measurement errors. To cope with this problem, this dissertation investigated a methodology that requires only a single endoscopic image, without any need for a reference, in order to estimate the size of an object of interest *in vivo*.

Another research direction investigated in this dissertation has led to the development of a novel approach for 3D model reconstruction, using an implicit neural representation with a periodic and parametric activation, named WaveShaping function. This function is utilized by a multi-layer perceptron (MLP) trained to learn a continuous function given a finite number of data points that describes a coarsely retrieved 3D model. Then, the MLP is regarded as a continuous implicit representation of that model; hence, it can interpolate data points to refine and restore regions of the 3D model at a higher resolution.

Recently, the interpretation of the inference process of deep learning models has received a lot of attention by the scientific community. Recent legislation and guidelines make the black box nature of these models unsuitable for commercial use. To tackle this problem, this dissertation proposes a novel, generalized framework for instantiating inherently interpretable CNN models, named E Pluribus Unum Interpretable CNN (EPU-CNN). An EPU-CNN model consists of CNN sub-networks, each of which receives a different representation of an input image expressing a perceptual feature, such as color or texture. The output of an EPU-CNN model consists of the classification prediction and its interpretation, in terms of relative contributions of perceptual features in different regions of the input image. EPU-CNN models have been extensively evaluated on natural and biomedical images concerning both binary and multiclass problems.

The various methodologies introduced in this dissertation outperform the respective state-of-the-art methods, while they are able to cope with various challenges that have been documented in the literature for each domain of application. Considering the saliency prediction, the proposed methods tackle the problem of estimating visual saliency both on biomedical and natural images. In addition, the salient object detection approach provides a novel solution aiming at reducing the dependency of such methods on additional sensors. The obstacle detection approaches effectively combine depth and visual saliency information to provide a reliable method for detecting obstacles in outdoor environments. A user-based evaluation of these methods showed that they can be efficiently incorporated into assistive navigation systems and aid visually impaired people to navigate outdoors. Regarding the single-image visual measurements proposed in this dissertation, the obtained results suggest that they can be successfully employed for measuring objects in everyday and medical applications. The use of such methods can aid towards the simplification of respective systems that require multiple sensors to perform such measurements. The use of implicit neural representations for the refinement of 3D models results in more accurate 3D representations of object that can be incorporated into digital twin models for *in-silico* clinical trials. Finally, the EPU-CNN framework satisfies the need for a generalized method for the construction of perceptually interpretable models. Additionally, EPU-CNN provides a way to cope with the requirements imposed by current legislations regarding the commercial applicability of ML

models. The research landscape explored by this dissertation is broad, and its contributions are expected to have both a societal and scientific impact, opening novel perspectives for further future research and the progress of science.

ΠΕΡΙΛΗΨΗ

Η μηχανική μάθηση (ML) είναι η βάση της μηχανικής αντίληψης και της υπολογιστικής όρασης. Η μηχανική αντίληψη αναφέρεται στην ικανότητα ενός υπολογιστικού συστήματος να κατανοεί διάφορες πτυχές του κόσμου μετά από επεξεργασία και ανάλυση δεδομένων που δέχεται ως είσοδο. Η μηχανική αντίληψη μέσω οπτικών δεδομένων υλοποιεί την έννοια της υπολογιστικής όρασης. Οι επιστημονικές συνεισφορές της παρούσας διδακτορικής διατριβής καλύπτουν ένα ευρύ φάσμα μεθόδων και εφαρμογών στο συγκεκριμένο πλαίσιο. Σε αυτές περιλαμβάνονται η ανίχνευση και ο εντοπισμός σημαντικών αντικειμένων, η ανίχνευση και η αποφυγή εμποδίων βάσει οπτικής πληροφορίας, οι οπτικές μετρήσεις με χρήση μίας εικόνας, η βελτίωση των τρισδιάστατων αναπαραστάσεων αντικειμένων και τα αντιληπτικά ερμηνεύσιμα μοντέλα μηχανικής μάθησης.

Η πρόβλεψη της οπτικής προσοχής σε ιατρικές εικόνες είναι ένα ερευνητικό θέμα που δεν έχει μελετηθεί ευρέως. Για να αντιμετωπιστεί αυτό το πρόβλημα, προτείνεται μια καινοτόμα μεθοδολογία εκτίμησης της οπτικής προσοχής των ιατρών, χρησιμοποιώντας Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks, CNNs) μαζί με μία νέα συνεργατική μέθοδο εκπαίδευσης, καθώς δημιουργήθηκε και ένα νέο σύνολο δεδομένων βάση της οπτικής προσοχής των ιατρών. Η προσέγγιση της οπτικής προσοχής των ανθρώπων αποτελεί δημοφιλές ερευνητικό θέμα- ωστόσο, ορισμένες από τις προσεγγίσεις με τις καλύτερες επιδόσεις απαιτούν τόσο έγχρωμη πληροφορία όσο και πληροφορία βάθους προερχόμενη από εξειδικευμένους αισθητήρες. Η ανάγκη για πρόσθετους αισθητήρες προσθέτει επιπλέον βαθμούς πολυπλοκότητας σε ένα σύστημα περιορίζοντας την εφαρμογή του. Στην παρούσα διατριβή, διερευνάται ο αντίκτυπος της ακριβούς εκτίμησης του βάθους για την αντίληψη της σημαντικότητας, με αποτέλεσμα ένα νέο σύστημα ανίχνευσης σημαντικών αντικειμένων βάση μίας εικόνας (MonoSOD), βασισμένο σε μια αρχιτεκτονική αυτόματου κωδικοποιητή CNN δύο κλάδων, ικανό να προβλέπει χάρτες βάθους και να εκτιμά το saliency.

Ακόμη μία συνεισφορά της παρούσας διατριβής είναι μια νέα μεθοδολογία για την ανίχνευση εμποδίων με βάση εικόνες RGB-D. Αυτή η μεθοδολογία ενσωματώνει αποτελεσματικά την ασαφή λογική και την πρόβλεψη της ανθρώπινης οπτικής προσοχής με τη χρήση Γενετικών Γενεσιουργών Δικτύων (Generative Adversarial Networks, GANs). Αυτός ο συνδυασμός μπορεί να μεταφράσει τη θέση των ανιχνευόμενων εμποδίων σε περιγραφικές γλωσσικές κωδικοποιήσεις που μπορούν να χρησιμοποιηθούν σε διάφορες εφαρμογές, όπως η ρομποτική και η υποβοηθούμενη πλοήγηση. Αναπτύχθηκε ένα μοντέλο CNN με αυτό-επιβλεπόμενη εκπαίδευση, με την ικανότητα προσομοίωσης της εν λόγω μεθόδου ανίχνευσης εμποδίων με βάση το RGB-D, με τη χρήση όμως μίας μόνο έγχρωμης εικόνας ως είσοδο.

Οι περιορισμοί των σημερινών μεθόδων οπτικής μέτρησης, όπως οι απαιτήσεις τους όσον αφορά τον αριθμό των εικόνων εισόδου και την ύπαρξη αντικειμένων αναφοράς, αποτέλεσαν το κίνητρο για την ανάπτυξη μιας καινοτόμου, πιο αξιόπιστης μεθόδου για την οπτική μέτρηση μεγέθους μίας εικόνας, η οποία ονομάζεται Virtual Grid Mapping (VGM). Η προτεινόμενη μέθοδος VGM απαιτεί μόνο μία εικόνα ως είσοδο και δεν απαιτεί καμία προηγούμενη πληροφορία σχετικά με τη σκηνή, όπως η γραμμή του ορίζοντα ή τα αντικείμενα αναφοράς. Ένα σημαντικό

πλεονέκτημα της VGM που συμβάλλει στην ευρωστία της είναι ότι αντιμετωπίζει την αβεβαιότητα που προέρχεται από τη βαθμονόμηση και την θέση της κάμερας, προσφέροντας έτσι μεγαλύτερη ακρίβεια από τις τρέχουσες μεθόδους μέτρησης οπτικού μεγέθους. Δεδομένων των γεωμετρικών ιδιοτήτων της κάμερας, η VGM παράγει και προβάλλει αυτόματα ένα πλέγμα εικονικών τρισδιάστατων σημείων στο δισδιάστατο επίπεδο της εικόνας, επιτρέποντας τη δημιουργία προσεγγιστικών αντιστοιχιών μεταξύ τρισδιάστατων σημείων του πραγματικού κόσμου και δισδιάστατων σημείων του επιπέδου εικόνας. Αυτές οι αντιστοιχίες επιτρέπουν την εκτίμηση της απόστασης μεταξύ ενός αντικειμένου και της κάμερας και στη συνέχεια τη μέτρηση του εν λόγω αντικειμένου. Για την εκτίμηση του μεγέθους των ανωμαλιών σε βιοϊατρικές εικόνες του γαστρεντερικού σωλήνα προτείνεται μια παρόμοια προσέγγιση με τη VGM που υποβοηθείται από το προβλεπόμενο βάθος που παρέχεται από ένα μοντέλο CNN. Διάφορες μελέτες επιβεβαίωσαν ότι η ενδοσκοπική εκτίμηση του μεγέθους των βλαβών έχει εγγενείς περιορισμούς και σημαντικά σφάλματα μέτρησης. Για να αντιμετωπιστεί αυτό το πρόβλημα, η παρούσα διατριβή διερεύνησε μια μεθοδολογία που απαιτεί μόνο μια ενδοσκοπική εικόνα προκειμένου να εκτιμηθεί το μέγεθος ενός αντικειμένου ενδιαφέροντος in vivo.

Μια άλλη ερευνητική κατεύθυνση που διερευνήθηκε σε αυτή τη διατριβή οδήγησε στην ανάπτυξη μιας νέας προσέγγισης για την ανακατασκευή τρισδιάστατου μοντέλου, χρησιμοποιώντας μια εσωτερική νευρωνική αναπαράσταση (Implicit Neural Representation, INR) με περιοδική και παραμετρική ενεργοποίηση, η οποία ονομάζεται WaveShaping function. Η συνάρτηση αυτή χρησιμοποιείται από ένα πολυστρωματικό νευρωνικό δίκτυο (Multilayer Perceptron, MLP) που εκπαιδεύεται για να μάθει μια συνεχή συνάρτηση δεδομένου πεπερασμένου αριθμού σημείων που περιγράφει ένα ατελές ανακτημένο τρισδιάστατο μοντέλο. Στη συνέχεια, το MLP θεωρείται ως μια συνεχής αναπαράσταση αυτού του μοντέλου- ως εκ τούτου, μπορεί να παρεμβάλει σημεία δεδομένων για να βελτιώσει και να αποκαταστήσει περιοχές του τρισδιάστατου μοντέλου σε υψηλότερη ανάλυση.

Πρόσφατα, η ερμηνεία της διαδικασίας εξαγωγής συμπερασμάτων των μοντέλων βαθιάς μάθησης έχει λάβει μεγάλη προσοχή από την επιστημονική κοινότητα. Το πρόσφατο νομοθετικό πλαίσιο για τη χρήση αλγορίθμων μηχανικής μάθησης καθιστούν τη φύση του μαύρου κουτιού που χαρακτηρίζει αυτά τα μοντέλα ακατάλληλη για εμπορική χρήση. Για την αντιμετώπιση αυτού του προβλήματος, η παρούσα διατριβή προτείνει ένα νέο, γενικευμένο πλαίσιο για την υλοποίηση εγγενώς ερμηνεύσιμων μοντέλων CNN, το οποίο ονομάζεται E Pluribus Unum Interpretable CNN (EPU-CNN). Ένα μοντέλο EPU-CNN αποτελείται από υποδίκτυα CNN, καθένα από τα οποία λαμβάνει μια διαφορετική αναπαράσταση μιας εικόνας εισόδου που εκφράζει ένα αντιληπτικό χαρακτηριστικό, όπως το χρώμα ή την υφή. Η έξοδος ενός μοντέλου EPU-CNN αποτελείται από την πρόβλεψη ταξινόμησης και την ερμηνεία της, από την άποψη της σχετικής συνεισφοράς των αντιληπτικών χαρακτηριστικών σε διαφορετικές περιοχές της εικόνας εισόδου. Τα μοντέλα EPU-CNN έχουν αξιολογηθεί εκτενώς σε φυσικές και βιοϊατρικές εικόνες όσον αφορά προβλήματα τόσο δυαδικών όσο και πολλαπλών κατηγοριών.

Οι διάφορες μεθοδολογίες που παρουσιάζονται στην παρούσα διατριβή υπερτερούν έναντι των αντίστοιχων σύγχρονων μεθόδων τεχνολογίας αιχμής, ενώ είναι σε θέση να αντιμετωπίσουν διάφορες προκλήσεις που έχουν καταγραφεί στη βιβλιογραφία για κάθε τομέα εφαρμογής.

Λαμβάνοντας υπόψη την πρόβλεψη της σημαντικότητας, οι προτεινόμενες μέθοδοι αντιμετωπίζουν το πρόβλημα της εκτίμησης της οπτικής προσοχής τόσο σε βιοϊατρικές όσο και σε φυσικές εικόνες. Επιπλέον, η προσέγγιση ανίχνευσης εμφανών αντικειμένων παρέχει μια νέα λύση με στόχο τη μείωση της εξάρτησης των εν λόγω μεθόδων από πρόσθετους αισθητήρες. Οι προσεγγίσεις ανίχνευσης εμποδίων συνδυάζουν αποτελεσματικά τις πληροφορίες βάθους και οπτικής προσοχής για να παρέχουν μια αξιόπιστη μέθοδο ανίχνευσης εμποδίων σε εξωτερικά περιβάλλοντα. Μια αξιολόγηση αυτών των μεθόδων με βάση τον χρήστη έδειξε ότι μπορούν να ενσωματωθούν αποτελεσματικά σε συστήματα υποβοηθητικής πλοήγησης και να βοηθήσουν τα άτομα με προβλήματα όρασης να πλοηγηθούν σε εξωτερικούς χώρους. Όσον αφορά τις οπτικές μετρήσεις μίας εικόνας που προτείνονται στην παρούσα διατριβή, τα αποτελέσματα που προέκυψαν υποδηλώνουν ότι μπορούν να χρησιμοποιηθούν με επιτυχία για τη μέτρηση αντικειμένων σε καθημερινές και ιατρικές εφαρμογές. Η χρήση τέτοιων μεθόδων μπορεί να βοηθήσει προς την κατεύθυνση της απλούστευσης αντίστοιχων συστημάτων που απαιτούν πολλαπλούς αισθητήρες για την εκτέλεση τέτοιων μετρήσεων. Η χρήση εσωτερικών νευρωνικών αναπαραστάσεων για τη βελτίωση των τρισδιάστατων μοντέλων οδηγεί σε ακριβέστερες τρισδιάστατες αναπαραστάσεις αντικειμένων που μπορούν να ενσωματωθούν σε μοντέλα ψηφιακών διδύμων για in-silico κλινικές δοκιμές. Τέλος, το πλαίσιο EPU-CNN ικανοποιεί την ανάγκη για μια γενικευμένη μέθοδο για την κατασκευή αντιληπτικά ερμηνεύσιμων μοντέλων. Επιπλέον, το EPU-CNN παρέχει έναν τρόπο αντιμετώπισης των απαιτήσεων που επιβάλλονται από τις τρέχουσες νομοθεσίες σχετικά με την εμπορική δυνατότητα εφαρμογής των μοντέλων μηχανικής μάθησης. Το ερευνητικό τοπίο που διερευνά η παρούσα διατριβή είναι ευρύ και οι συνεισφορές της αναμένεται να έχουν τόσο κοινωνικό όσο και επιστημονικό αντίκτυπο, ανοίγοντας νέες προοπτικές για περαιτέρω μελλοντική έρευνα και την πρόοδο της επιστήμης

To my family

ACKNOWLEDGMENTS

A.I. has took the world at storm recently, with advancements in the field that indicate that we currently living through the beginning of a new era. During my doctoral studies I was lucky to be in a place that I could observe, study and contribute to this extraordinary field of machine learning. One of the most valuable lessons I've learned during this research journey is that in a fast-paced everchanging field, you don't need to know everything to push the boundaries; but you absolutely need to know how learn.

First and foremost, I would like to express my heartfelt gratitude to my parents and sister for their unwavering support and encouragement throughout these years. They were always by my side and helped me with all means possible for which I will be forever grateful. I would also like to thank my supervisor and mentor Prof. Dimitris Iakovidis for introducing me and guiding me through the fascinating world of scientific research. His advices, in depth knowledge of machine learning and computer vision and deep sense of scientific rigor have been inspiring and determinant factors for my dedication and progress in my doctoral studies.

I would like to thank my partner and friend Ioanna Asanakidi for supporting me through the highs and lows that one faces during the doctoral studies, as well as for the interdisciplinary conversations that provided a different perspective to various research topics. I would also like to thank my all my colleagues at the lab and especially Dr. Panagiotis Kalozoumis, Dr. Dimitris Diamantis and Dr. Michael Vasilakakis for the insightful conversations, sharing of ideas and fun times. No matter the time and place, if you share it with great people, you always have fun.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Machine learning (ML) is a scientific discipline, strongly related to computer science, that recently overwhelmingly attracts the interest of academics, the industry and the general public. The last decade, the advancements of electrical engineering and computer science (*e.g.*, the development of high-performance Graphical Processing Units, GPUs, the development of easy-to-use machine learning frameworks, *etc.*) have made the utilization of ML algorithms faster, easier, and more reliable. As a result, nowadays ML has flooded all kinds of domains including art (Ndou, Ajoodha, & Jadhav, 2021), linguistics (Otter, Medina, & Kalita, 2020) as well as high-risk fields such as autonomous driving (Mozaffari, Al-Jarrah, Dianati, Jennings, & Mouzakitis, 2020) and medicine (Piccialli, Di Somma, Giampaolo, Cuomo, & Fortino, 2021). Historically, the origin of machine learning as we know it is usually associated with the psychologist Frank Rosenblatt (Fradkov, 2020). Inspired by the human nervous system, Rosenblatt developed an automaton, named "perceptron", purposed to recognize letters of the alphabet (Rosenblatt, 1957). This primitive perceptron discretized analog signals via a thresholding element and utilized them to perform its recognition task. Rosenblatt was the first to perform initial mathematical studies regarding the capabilities and training conditions of the perceptron that were later gain more exposure through the Novikoff theorem (Novikoff, 1963; Rosenblatt, 1958, 1960). The research and funding for the investigation of the capabilities and prospects of the perceptron however have been reduced when a book published in 1969 by Minsky and Papert emphasized some of its limitations regarding its capacity to represent logical functions like XOR and NXOR (Fradkov, 2020; Minsky & Papert, 2017).

Nevertheless, the scientific community did not lose its interest in researching to advance learning algorithms. Researchers kept investigating the learning abilities of multilayer neural networks and the first multilayer convolutional neural network, named Neocognitron, has been proposed in 1982

(Fukushima & Miyake, 1982). A huge impact to the ML domain has been made with the rework of the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1985). The backpropagation enabled the efficient training of learning algorithms and introduced several benefits over previous approaches. A significant breakthrough was achieved with the introduction of support-vector-networks that heavily impacted the ML community (Mohri, Rostamizadeh, & Talwalkar, 2018). Since then, the advancement of ML algorithms has been constant, initially led by Geoffrey Hinton and followed by the triad Yann LeCun, Yoshua Bengio and Juergen Schmidhuber who set the foundations for the modern deep learning (DL) (LeCun, Bengio, & Hinton, 2015; Schmidhuber, 2015).

Deep learning has penetrated and revolutionized the way problems are handled almost in every machine learning application. In natural language processing initially Long Short-Term Memory networks (Schmidhuber, Hochreiter, & others, 1997) and later transformers (Wolf et al., 2020) provided exceptional results with a recent model being capable to even write a scientific report about itself (Thunström & Steingrimsson, 2022). In computer vision Convolutional Neural Networks change the way the problems were tackled by alleviating the requirement for manual feature extraction and making tasks that seem extremely hard, such as novel image generation, a reality (Ramesh, Dhariwal, Nichol, Chu, & Chen, 2022). In 3D modeling and physics, DL models have been proposed that are capable of synthesizing novel views of complex scenes and interpolating experiments to provide detailed insight in a given problem (S. Cai, Mao, Wang, Yin, & Karniadakis, 2022; Mildenhall et al., 2020). In medicine, CNNs and other ML models can help physicians with tedious tasks such as long video examination or assist them in their diagnosis procedure (Anwar et al., 2018; Dimitrios K Iakovidis, Tsevas, & Polydorou, 2010). The list of deep learning applications that can augment the problem solving in various fields is endless and new approaches emerging every day.

Regardless of their exceptional performance, deep learning models have some fundamental drawbacks that prevent their applicability in real-world applications and domains of high-risks. First off, recently the commercial applicability of Machine Learning (ML) algorithms has been regulated through legislation acts (Selbst & Powles, 2018) that aim at making the world ‘fit for the digital age’ (Steponnait, 2016) with requirements, safeguards, and restrictions regarding ML and automatic decision-making in general. A crucial aspect regarding the compatibility of ML models concerning these regulations is interpretability. But how is the interpretability of ML models defined? According to the recent literature (Arrieta et al., 2020), interpretability refers to a passive characteristic of a model, indicating the degree to which a human understands the cause of its decision. Hence, the provided interpretations of the decision-making process of a model can limit its opaqueness (Castelvecchi, 2016) and earn users’ trust, e.g., by offering interpretations for risk-sensitive decisions in medicine. In real-world tasks, the discriminative power of ML models, as

expressed by their performance measures, e.g., their predictive accuracy, is regarded as an insufficient descriptor of their decisions (Rudin et al., 2022).

Then the data required for training such models limits their application to domains that publicly available datasets are in abundance, or the data acquisition is hard. For example, research in size measurements and gaze estimation in medical images is difficult since there are not many datasets available that can be used (George Dimas, Iakovidis, & Koulaouzidis, 2019; Dimitris K Iakovidis et al., 2021).

The main focus of this dissertation is the investigation and development of novel deep learning models in the context of machine perception and computer vision. The models that are proposed and investigated in this dissertation have the capacity to perceive their surroundings in terms of regions of interest such as salient objects, obstacles *etc.*, as well as being capable of grasping the essence of 3D space. An additional essential subject that this thesis investigates is the problem of understanding how a model infers a particular result. For this purpose, models that have the capacity to provide reasoning regarding to their results with respect to perceptual information that can be easily understood by humans are studied. The results of the models that are presented in this dissertation indicate that these models can be employed efficiently for the tasks of *a)* detecting salient object and regions based on human perception on natural and biomedical images; *b)* obstacle detection for the assistive navigation of people with visual impairments and robots; *c)* measuring the size of real-world objects as well as abnormalities in biomedical images; *d)* perceiving, representing and reconstructing 3D objects in high quality from sparse 3D representations; and *e)* generalized interpretable classification of natural and biomedical images with perceptually human-understandable explanations.

1.1 Aims of this Dissertation

The research focus of this dissertation is the development of novel ML approaches in the context of machine perception and computer vision. The methodologies that emerged have broad social impact and they can be summarized as follows:

- The investigation of saliency estimation in natural images as well as biomedical images based on human perception using less generated multimodal data.

- The investigation of decision-making approaches in the context of obstacle detection and avoidance for computationally assistive navigation.

- The investigation of size measurements methodologies that use a single image (monocular) to assess the dimension of a target object tested on natural images of publicly available datasets.

- The investigation of methods for the 3D reconstruction of object from sparse point cloud representations

- The investigation of interpretable DL models that can provide perceptual explanations about their results in accordance with human perception that can be used for signal analysis

1.2 Thesis Contributions

The research performed to tackle the aims that were set for this dissertation resulted in the development of novel DL frameworks, methodologies, and applications. Specifically, the results that emerged lead to the publication of six (7) articles in scientific journals with a review process from which five (5) are published and two (2) are under review; eleven (11) scientific papers in proceedings of international conferences from which ten (10) are published and one (1) under review. The contributions of this dissertation are summarized as follows:

- A CNN model and training scheme for the estimation of gaze patterns of a physician in images of the gastrointestinal tract (George Dimas, Iakovidis, et al., 2019; Gatoula, Dimas, Iakovidis, & Koulaouzidis, 2021).

- A CNN model that can detect salient object based on RGB images and complementary depth information generated by a Generated Adversarial Network (GAN) (George Dimas, Gatoula, & Iakovidis, 2021).

- An obstacle detection approach based on gaze estimation and fuzzy logic aiming to assist people with visual impairments for their navigation in outdoor settings (George Dimas, Diamantis, Kalozoumis, & Iakovidis, 2020; George Dimas, Ntakolia, & Iakovidis, 2019; Dimitris K Iakovidis, Diamantis, Dimas, Ntakolia, & Spyrou, 2020).

- A self-supervised CNN model that can efficiently detect obstacles in outdoor settings without the need for additional depth information (George Dimas, Cholopoulou, & Iakovidis, 2021).

- A single-image methodology for *in-vivo* measurements (George Dimas, Bianchi, et al., 2020).

- A single-image methodology for the measurement of the dimensions of objects in natural images (George Dimas & Dimitris Iakovidis, 2022).

- An unsupervised methodology for the reconstruction of 3D models, given a sparse point cloud representation (P. Kalozoumis, Dimas, Triantafyllou, & Iakovidis, 2022; Triantafyllou, Dimas, Kalozoumis, & Iakovidis, 2022).

- A generalized framework for the development of interpretable CNNs that can provide interpretations in accordance with human perception (George Dimas, Cholopoulou, & Iakovidis, 2022).

1.3 Thesis Outline

The rest of this thesis is organized in five (7) chapters:

- Chapter 2 summarizes the theoretical background regarding machine perception and computer vision including image understanding and deep learning processes.

- Chapter 3 presents the methodologies that were developed for predicting visual saliency on images.

- Chapter 4 presents the methodologies for obstacle detection.

- Chapter 5 presents the methodologies that were developed for scene perception, *i.e.*, measurement methodologies based on a single image.

- Chapter 6 introduces a novel methodology for the reconstruction of coarse 3D models using ML.

- Chapter 7 introduces approaches that investigate methodologies that develop models that can perceive their environment in a perceptually interpretable way.

- Chapter 8 presents the conclusions of this dissertation, and directions for future work.

Chapter 2
Visual Sensing & Perception

Machine Perception (MP) can be defined as the ability of the machine to process input data, *e.g.*, visual, acoustic, tabular *etc.*, to understand various aspects of the world (Parasher, Sharma, Sharma, & Gupta, 2011). The development of intelligent systems that have the capacity to perceive the world through visual data, *e.g.*, images, is defined as Computer Vision (CV). In CV, the machine processes data extracted from images either in the form of features engineered by humans (like in traditional ML) or through a deep learning paradigm where the machine figures out features on its own, through a learning process (that we refer to as training), that benefits its world comprehension (Shapiro, Stockman, & others, 2001).

This chapter summarizes the required theoretical background regarding *a)* camera models and image acquisition, *i.e.*, how a sensor captures and represent a scene in a digitized form, *b)* the fundamentals of intelligent machines such as Multilayer Perceptrons (MLP) and Convolutional Neural Networks (CNNs), *c)* how the intelligent machines are trained, *d)* methodologies that enable the understanding of how these machines perceives their environment and *e)* novel applications on how to exploit the properties of neural networks as global function approximators for data representation.

2.1 Imaging with Visual Sensors

Light is the fundamental element of photography, and its nature makes photography a major chapter of physics. The formation of an image as captures by a camera, can be described by employing geometry in the field of optics. A major physical principle that makes photography possible is the photoelectric effect. By leveraging both the photoelectric effect and geometry regarding the emission of light rays, we can create cameras that capture and visualize a scene at a particular moment in time. Then, these instantiations of a scene, known as images, can be processed by intelligent machines and enable them to understand their surroundings or assist humans in decision making. The focus of this chapter will be to provide the required background regarding the camera models and their geometric properties that makes photography possible in various scenarios. In addition, the basic sensors that exist for capturing images that represent different energies of light or information regarding the scene (*e.g.*, color, infrared, depth) will be described.

2.1.1 Mathematical Modeling of Camera Systems

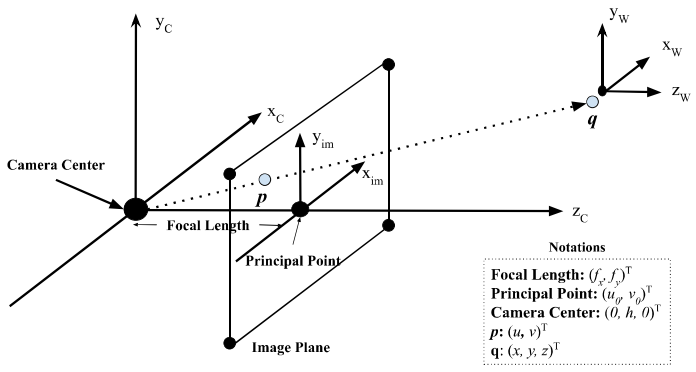


Figure 2.1. Illustration of the pinhole camera model.

A camera model is described by two set of basic parameters, the extrinsic and intrinsic parameters. The extrinsic parameters are used in a procedure which transforms the coordinates of the scene that is going to be projected in an image, to a camera centered coordinate system (frame). The intrinsic parameters of the camera include the focal length, a scale factor and the image center which is also described in the literature as principal point (Heikkila & Silvén, 1997). There are also other set of parameters that enable a more detailed mathematical expression of the camera model such as, the distortion coefficients that model the distortion that may be introduced to an image based on the lens of the camera. A well-known and widely used mathematical model of a camera is the Pinhole Camera Model. The pinhole camera model is based on the principle of collinearity, where each point in the space of the 3D scene in projected by a straight line through the projection center (principal point) into the 2D image plane (Figure 2.1).

The origin of coordinate system of the camera is at the projection point of the camera and it is denoted as X_C, Y_C, Z_C with the z axis being perpendicular to the image plane as illustrated in Figure 2.1. The coordinate system of the 3D plane of the scene is denoted as X_W, Y_W and Z_W. The rotation process is based on Euler angles ω, φ and κ which denote three sequential elementary clockwise rotation around x-, y- and z-axis, respectively. For the world coordinate system to match the

coordinate system of the image plane, the first rotation is performed with respect to x-axis, then with respect to y- and finally with respect to z-axis. After the rotation of the world axis, to precisely match the coordinate system of the image plane, the rotated coordinates need to be translated along all axis, with a displacement of Δx, Δy, Δz. Considering the above, for a point $P = (x, y, z)^T$ of the 3D world plane (scene) that is to be projected to its corresponding point the 2D image plane, this point P needs initially to be transformed as follows:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{34} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} \tag{2.1}$$

where each term corresponds to the following transforms:

$$m_{11} = \cos(\varphi)\cos(\kappa) \tag{2.2}$$
$$m_{12} = \sin(\omega)\sin(\varphi)\cos(\kappa) - \cos(\omega)\sin(\kappa) \tag{2.3}$$
$$m_{13} = \sin(\omega)\sin(\varphi)\cos(\kappa) + sin(\omega)\sin(\kappa) \tag{2.4}$$
$$m_{21} = \cos(\varphi)\sin(\kappa) \tag{2.5}$$
$$m_{22} = \sin(\omega)\sin(\varphi)\sin(\kappa) + \cos(\omega)\sin(\kappa) \tag{2.6}$$
$$m_{23} = \cos(\omega)\sin(\varphi)\sin(\kappa) - sin(\omega)\cos(\kappa) \tag{2.7}$$
$$m_{31} = -\sin(\varphi) \tag{2.8}$$
$$m_{32} = \sin(\omega)\cos(\varphi) \tag{2.9}$$
$$m_{33} = cos(\omega)\cos(\varphi) \tag{2.10}$$
$$\Delta x = -m_{11}X_C - m_{12}Y_C - m_{13}Y_C \tag{2.11}$$
$$\Delta y = -m_{21}X_C - m_{22}Y_C - m_{23}Y_C \tag{2.12}$$
$$\Delta z = -m_{31}X_C - m_{32}Y_C - m_{33}Y_C \tag{2.13}$$

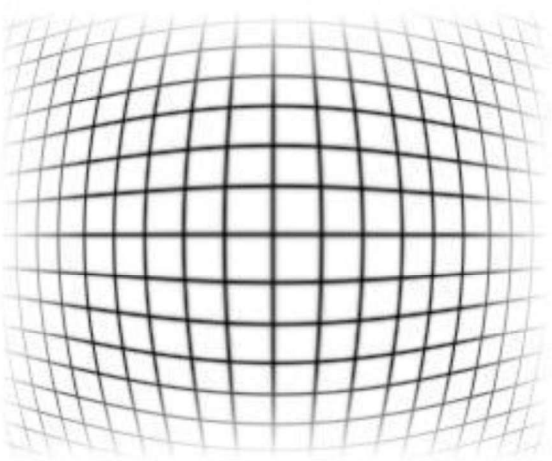
where Eq. (2.2-10) express the Euler angles. Once this transformation takes place, then the intrinsic parameters of the camera can be used for the finalization of the projection of the 3D point P to its corresponding point in the 2D image plane, p. The projection of the transformed $P' = (x', y', z')^T$ to the 2D image plane is performed as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{2.14}$$

Where f is the focal length of the camera. However, one more step is required to obtain the point P in pixel coordinates:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} D_u s_u u \\ D_v v \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{2.15}$$

s_u is a scale factor, D_u, D_v are coefficients enabling the transition from the metric unit system to pixels and u_0, v_0 denotes the principal point of the camera.



This generic mathematical representation of the pinhole camera model is not applicable everywhere, nevertheless, is used as the basis for more advanced models. Usually, the lens that cameras use introduce a distortion effect to the captured image. A common distortion that occurs is that of the radial distortion. This distortion causes the actual image points to be displaced radially in the image plane (Figure 2.2) (Slama, 1980). The effect of radial distortion on the points of an image can be approximated using the following formula:

Figure 2.2. Example of radial distrotion

$$\begin{pmatrix}\delta u_r{}'\\ \delta v_r{}'\end{pmatrix} = \begin{pmatrix}u'(k_1 r^2 + k_2 r^4 + \ldots)\\ v'(k_1 r^2 + k_2 r^4 + \ldots)\end{pmatrix} \tag{2.16}$$

where k_1, k_2, …, are the radial distortion coefficients whereas $r = (u'^2 + v'^2)^{1/2}$.

Tangential distortion is another regularly occurring distortion. This distortion is in effect when the centers of curvature of lens surface are not always strictly collinear and is characterized as decentering including both a radial and tangential component. Tangential distortion is modelled as follows:

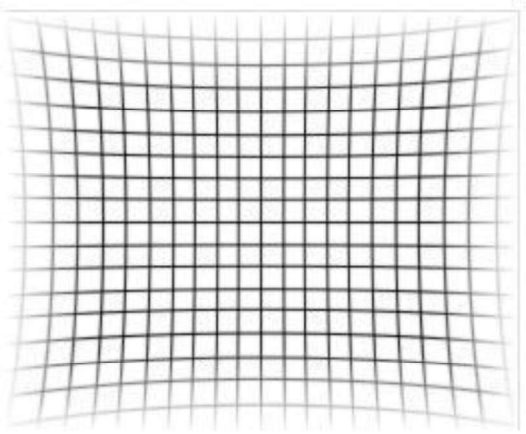


Figure 2.3. Example of tangential distortion

$$\begin{pmatrix}\delta u_t{}'\\ \delta v_t{}'\end{pmatrix} = \begin{pmatrix}2p_1 u'v' + p_2(r^2 + 2u^2)\\ p1(r^2 + 2v^2) + 2p_2 u'v'\end{pmatrix} \tag{2.17}$$

p_1 and p_2 are the coefficients of the tangential distortion. By combining Eq. (2.16) and Eq. (2.17) and incorporating them in the basic model of the pinhole camera, *i.e.*, Eq. (2.15), a more accurate formal expression of a camera is derived:

$$\begin{pmatrix}\tilde{u}\\ \tilde{v}\end{pmatrix} = \begin{pmatrix}D_u s_u(u + \delta u_r' + \delta u_t')\\ D_u(v + \delta v_r' + \delta v_t')\end{pmatrix} \tag{2.18}$$

2.1.2 Types of Imaging Sensors

There are many types of imaging sensors that are used in various camera systems. The camera sensor is responsible for collecting the visible or non-visible light converting it to an electrical signal that can be organized to render images and videos. Multiple camera sensors can be used in different arrangements to extract information from a scene such as the depth information. This section is dedicated to three different sensors that have been used and investigated in various application described in this dissertation.

RGB Sensor

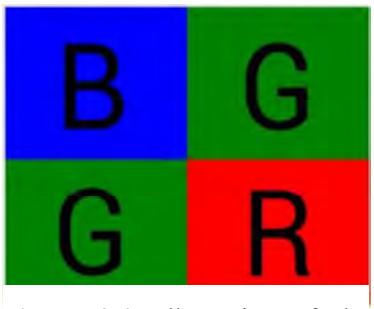


Figure 2.4. Illustration of the Bayer Filter Array BGGR pattern

An RGB sensor is responsible for converting visible light with a wavelength of 400 to 700nm to an electrical signal that can be used to generate an image or a sequence of images, *i.e.*, a video. RGB camera sensors are designed to perceive color in a similar way to the human eye. These sensors usually have cells equipped with 3 kinds of filter elements. Each one is capable of filtering a band of light isolating red, green and blue colors (Lukac, Plataniotis, & Hatzinakos, 2005). These cells along with the filter elements are structured in a Color Filter Array (CFA) with a Bayer BGGR pattern (Figure 2.10). The filter proportions, *i.e.*, 2 green filters, one red and one blue, have been chosen to mimic the human eyesight. The 2 green filters are justified because of the increased sensitivity of humans to the color green during daytime.

Each cell receives all color information but due to the filtering each cell receives only photons with a wavelength that corresponds to either red, green, or blue color. After the deciphering of the color information by each cell of the camera sensor, follows a process that is called demosaicing. This process is performed both by firmware of the camera and embedded software to enable the transformation of each Bayer pattern element of the sensor array to pixels and form the final image.

Stereo Camera Sensor

A single RGB camera is capable of capturing a scene in the form of an image. Although the color information of a scene is useful, it is not always sufficient in the context of computer vision applications. For example, for autonomous driving, color information is useful for detecting objects in the scene, categorizing them in classes or segmenting the image, based on its semantics. For the navigation of the car, however, a 3D perception of the scene is necessary. For this purpose, there are various approaches to assess 3D information, such as specialized sensors like sonar sensors, laser-based sensors (LIDAR) *etc*. Another approach is the usage of multiple cameras in a

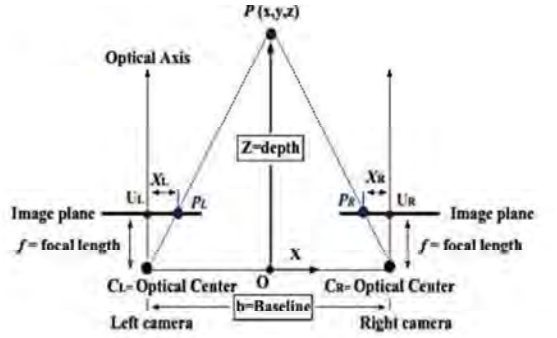


Figure 2.5. Example of a stereo camera

Figure 2.6. Example of a depth map

horizontal or vertical arrangement. With two cameras arranged in straight line vertically or horizontally, the distance to a point can be measured provided that the point is in the overlapping view of the two cameras. An illustration of a stereo setup is illustrated in Figure 2.5. By considering the triangles PC_LC_R and PP_LP_R the value of depth that denotes the distance z between the camera and the point that is depicted in an image at a pixel $p = (x, y)^T$ in a pair of images captured at the same time by the left and right camera can be calculated as follows:

$$z = \frac{b * f}{x_l - x_r} \tag{2.19}$$

where b is the distance between the optical centers of the two cameras measured in world units, *i.e.*, cm, mm *etc.*, the difference x_l - x_r is the portion known as disparity where x_l and x_r indicate the position of the coordinate x of p on the left and right image. The magnitude of the disparity is inversely proportional to the distance between p and the camera, *i.e.*, the bigger the disparity the closer the point to the camera. By applying Eq. (2.19) for each pixel p on an image captured by a stereo camera system, the estimated depth values z can be organized in an array with the same dimensions of the original image. This array, each element of which indicates the depth value of the scene, is called a depth map. An example of depth map captured by a stereo camera system is illustrated in Figure 2.6.

The use of multiple cameras of images for assessing 3D cues regarding a scene is called Stereo Vision. Stereo vision is currently used in many applications, ranging from navigating vehicles on another planet, to assisting people with visual impairments in their day to day lives (George Dimas, Ntakolia, et al., 2019; Scaramuzza & Fraundorfer, 2011).

Near Infrared Sensor

Conventional RGB cameras operate well in well-lit environments, during the daylight or by using artificial light in darker sceneries. There are cases, however, where during the daylight because of atmospheric conditions like the presence of fog, smoke *etc*. or not well-lit scenes, the performance of the cameras deteriorate. Hence, the applicability of conventional cameras in stereo vision setups is limited. The limitation that is introduced by conventional cameras lies to the fact that they exploit only a specific spectrum of the light that is known as visible light. There are other sensors that can produce images of a scene by leveraging the non-visible light like the infrared or near-infrared

Figure 2.7. Example of a near infrared image

Figure 2.8. Example of a near-infrared / RGB stereo vision camera

radiation. Such stereo vision systems usually incorporate both RGB and infrared sensors. A representative example is the RGB-D camera Intel D435. D435 utilizes the infrared based sensors for producing depth maps and RGB sensors for capturing the color information of the scene (Grunnet-Jepsen, Sweetser, Winer, Takagi, & Woodfill, 2018; Hussmann, Ringbeck, & Hagebeuker, 2008). Such camera systems provide robust imaging that is not affected by atmospheric or other conditions. Additionally, D435 uses projectors that emit optical patterns using the infrared spectrum to enhance the depth information that can be acquired. Such projectors overlay the observed scene with a semi-random texture that facilitates finding correspondences and are particularly useful in the case of texture-less surfaces like indoor dimly lit white walls. Since these optical patterns are emitted in the infrared spectrum, they do not interfere with the RGB sensor. As a result, the D435 can simultaneously capture color images and acquire accurate depth information.

2.2 Intelligent Machines

When we refer to intelligent machine, we mean systems that are capable of learning on their own through an algorithmic training process. Once they are trained, these systems are capable of providing us with estimations given a set of input data. This process is called inference. In detail, machine learning systems are capable of detecting patterns in large amounts of data when they are tasked to tackle a specific problem (Society, 2017). Another definition for machine learning models, that can be considered more detailed, is the automated process of discovering correlations between variables in a dataset, for the purpose of making predictions or estimates of some outcome (Lehr & Ohm, 2017).

Many machine learning models have been proposed in the literature, some of the most well-known being, Decision Trees, Naïve Bayes, Support Vector Machines, MLPs, CNNs *etc.* (Mahesh, 2020). This dissertation, however, focuses on MLPs and CNNs as the most notorious machine learning (and deep learning) models that dominated the CV scene. MLPs and CNNs have been successfully employed in various CV applications, from recognizing handwritten characters to the detection of abnormalities in biomedical images. The modeling for both ML algorithms has been based on

biological systems of the brain and the eye. In the following sections, more details are provided with respect to the modeling, training, and the inference process of these models.

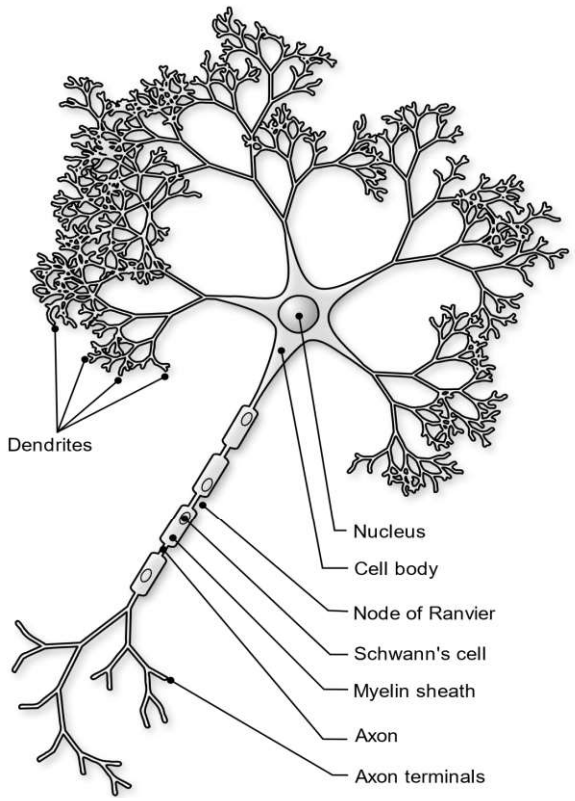


Figure 2.9 Illustration of a biological neuron and its various parts

2.2.1 The Perceptron (a.k.a. Artificial Neuron)

The fundamental block of MLPs, is called perceptron, also known as, artificial neuron. The design of the perceptron has been inspired by the neurons (nerve cells) of the brain. The biological neurons are the fundamental units of the brain and the human nervous system. These cells are responsible for the reception of signals originating from the external world as well as for providing responses that enable specific biological functions (*e.g.,* motor functions of muscles) and for transforming and relaying electrical signals from the reception of the signal to the final function. Approximately a human employs 100 billion neurons that they define its personality, the responses of the body and most of the aspects of the human being.

To understand how the design of the perceptron has been conceived we need to study the architecture of a biological neuron. One can imagine the structure of a biological neuron as a tree consisting of three main parts: *a)* dendrites, *b)* an axon and *c)* the main cell body (soma). By continuing the tree analogy, these three parts of the neuron correspond to the branches, roots and trunk of a tree, respectively. The dendrites are responsible for receiving the input signals that derive from other cells, the axon is responsible of providing a transformed version of the input signal while the soma is where the DNA of the neuron is stored in its nucleus and the proteins that are transported via the axon and the dendrites are synthesized. To better visualize the overall structure of a biological neuron an illustration of its structure is presented in Figure 2.9.

A perceptron can be considered as a simplified version of a biological neuron. Schematically, the graphical representation of a perceptron is very similar to that of a biological neuron. To better understand the similarities between the two, an illustration of an artificial neuron is presented in Figure 2.10. From the inspection of the two illustrations, *i.e.*, of the biological (Figure 2.9) and the artificial neuron (Figure 2.10) we can easily draw some correspondences between the two designs. The weights of the artificial neuron can be considered as the dendrites, since they receive an input

that is transformed by some weights. The combination of the transfer and activation function resembles the soma of the neuron where the input stimuli are collected by the dendrites

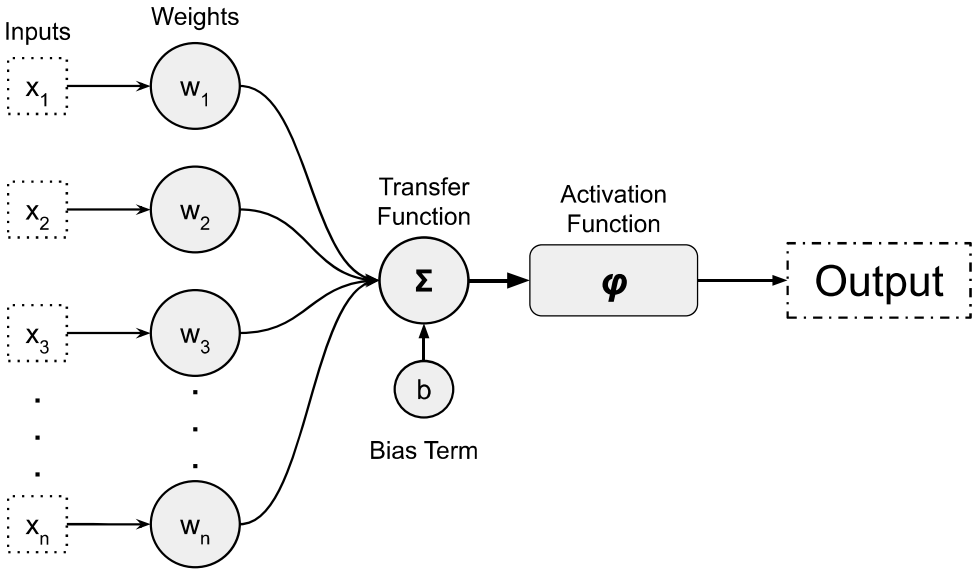


Figure 2.10 Illustration of a perceptron and its parts

to be propagated to the axon that outputs the final transformed signal. The perceptron is a parametric function that can be modeled mathematically as follows (Kubat & Kubat, 2017):

$$y = \varphi(b + W^T x) \tag{2.20}$$

or

$$y = \varphi\left(b + \sum_{i=1}^{n} w_i x_i\right) \tag{2.21}$$

where Eq. (2.20) is the mathematical model of a perceptron using linear algebra whereas Eq. (2.21) is its analytical form. In Eq. (2.20) $W = (w_1, w_2, w_3, \ldots, w_n)^T$ and $x = (x_1, x_2, x_3, \ldots, x_n)^T$ are the vectors corresponding to the weights of the perceptron and input stimuli, while b is a bias term. Function $\varphi(\cdot)$ is called an activation function and is usually used to introduce non-linearity to the perceptron making it capable to approximate complex non-linear functions.

Table 2.1 Example of criteria

Criteria	Weights
(x_1) *Weather is good*	$w_1 = 0.3$
(x_2) *I had a good meal 1 hour ago*	$w_2 = 0.7$
(x_3) *I have free time of >1 hour*	$w_3 = 0.6$

Let's use the mathematical model of a perceptron to examine some of its applications. The perceptron is used to approximate other function that their mathematical model is unknown; however, some examples of pairs of

inputs and outputs need to be known for training. Let us consider a perceptron that we want to decide for us whether we should work out or not. The inputs are a set of criteria that are either True or False encoded in a binary format of 0 and 1, respectively. Each criteria resembles an input x_i, $i = 1, 2, 3, \dots n$, as presented in Table 2.1. Each of these criteria is assigned with a value of 1 or 0 representing whether a statement is True or False. The next step is to assign to each criterion a weight w_i, $i = 1, 2, 3, \dots n$, to enable the decision-making process of the perceptron (Table 2.1). These weights derive from a training process during of which the perceptron converges to the best possible set of weights that provide the best possible outcome according to a particular task. In this example, it is assumed that the perceptron is already trained to provide the best decision regarding our workout. More details regarding the training process of MLPs and CNNs are provided in the following sections (see section 2.4). One more component that it needs to be determined is the activation function that according to Eq.(2.21), $\varphi(\cdot)$, outputs the final decision of the perceptron. In this case the φ will be a Heavyside function (*i.e.*, step function) parametrized by a:

$$\varphi(x; a) = \begin{cases} 1, x > a \\ 0, x \leq a \end{cases} \tag{2.22}$$

Table 2.2. Criteria and their respective exemplary values

Criteria	Values
(x_1) *Weather is good*	*0*
(x_2) *I had a good meal 1 hour ago*	*1*
(x_3) *I have free time of >1 hour*	*0*

where a for this example is set to $a = 0.9$. Let's set a scenario where the criteria presented in Table 2.1 have the values presented in Table 2.2. For simplicity the bias term is set to 0. Then according to Eq. (2.20), by considering the respective values and weights of the criteria the perceptron proceeds to the following calculations presented in Figure 2.11. As it can be observed,

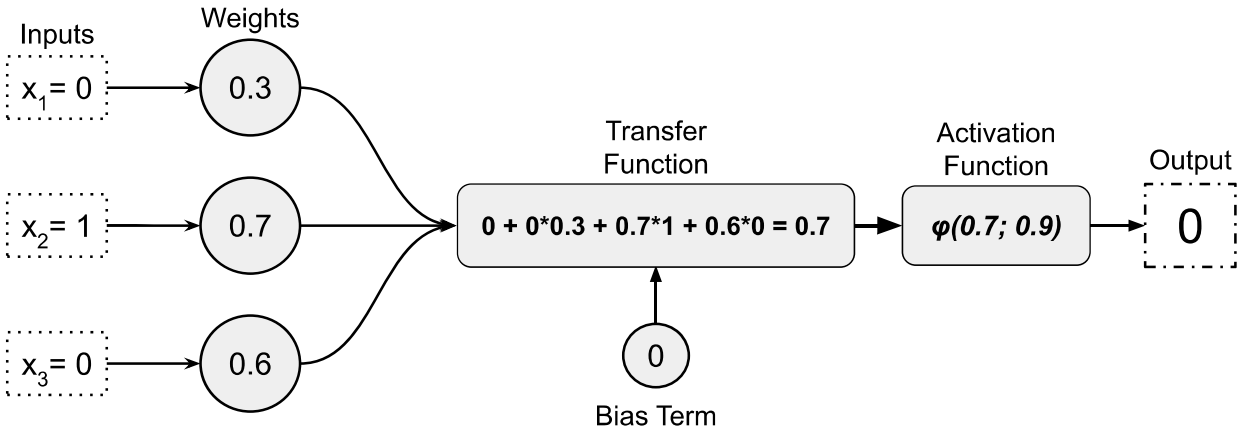


Figure 2.11. Illustrations of the calculations made by a perceptron for the example described in Section 2.2.1.

given the weights of the perceptron and the values that have been provided as input, its prediction is 0, *i.e.*, the user should not workout.

However, the perceptron model is only capable of solving classification problems where the input vectors are linearly separable. A common problem that is considered unsolvable by a simple perceptron is the approximation of the XOR logical gate. How are such problems tackled? Considering that our brain does not use neurons in a single layer but networks of multiple layers of neurons connected to each other, more sophisticated networks have been proposed that can approximate any function. These universal approximators are known as Multilayer Perceptrons or Multilayer Neural Networks (Hornik, Stinchcombe, & White, 1989).

2.2.2 The Multilayer Perceptron (a.k.a. Artificial Neural Network)

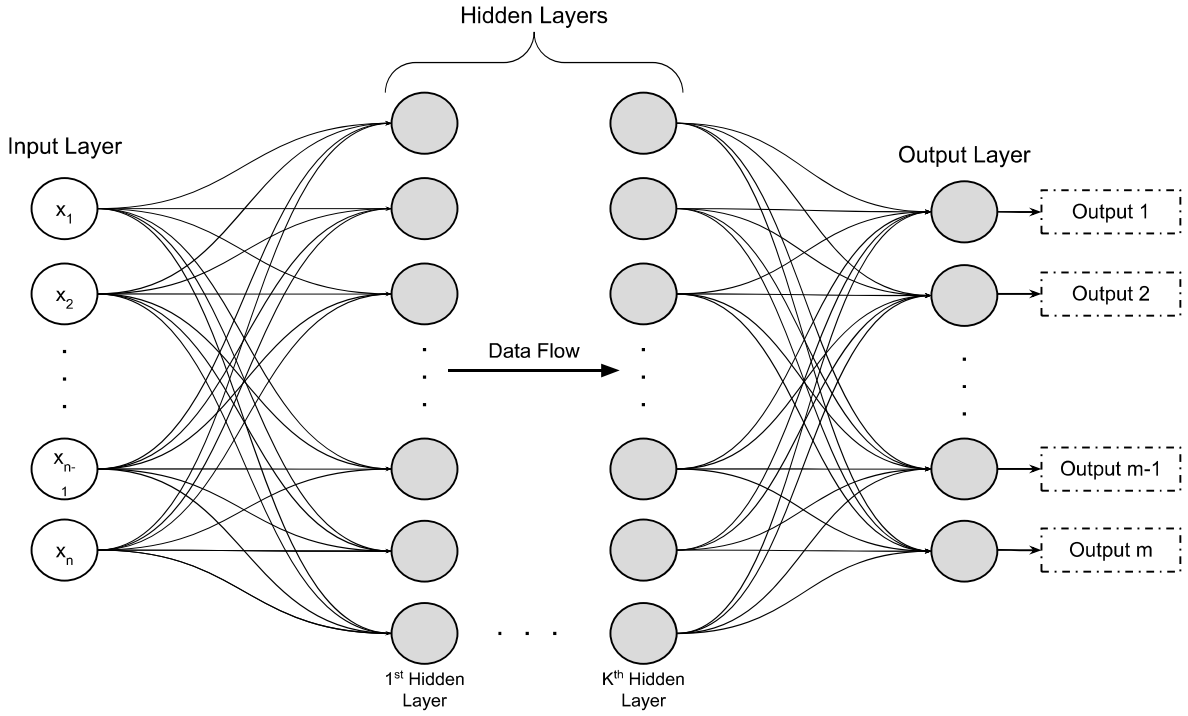


Figure 2.12. Illustration of Multilayer Perceptron. Gray circles denote artificial neurons, white circles input data that are not processed by artificial neurons and the connections among neurons are illustrated with black lines.

The perceptron has been modeled to mimic the structure and functionality of the biological neurons that can be found in the brain. The biological brain, however, is far more advanced, consisting of billions of neurons that are connected in a complex way. The simple perceptron model is not sufficient to emulate the more complicated design of the brain, and as a result it is limited in solving only problems that incorporate linearly separable set of vectors.

A more advanced model is the Multilayer Perceptron (MLP) also known as Artificial Neural Network (ANN). Instead of utilizing a single layer of Perceptrons, an MLP consists of multiple

layers each of which is fully connected with the layer that follows. An illustration of the MLP architecture is illustrated in Figure 2.12. As it can be observed in Figure 2.12, an MLP consists of at least three layers of artificial neurons. The first layer is called input layer, the intermediate layers of the MLP are called hidden layers and the final layer that provides the desired output is named output layer. Each unit of the first layer corresponds to the elements of the input vector

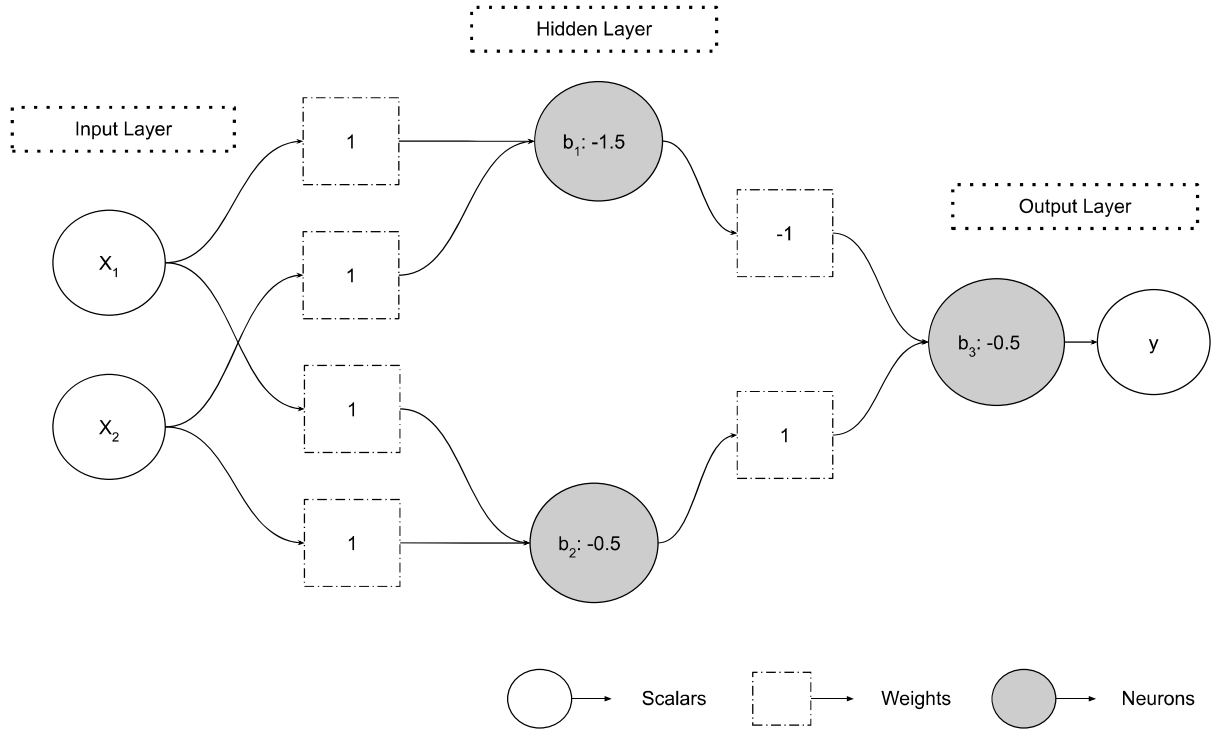


Figure 2.13 Illustration of a three-layer MLP trained to approximate the XOR logic gate.

whereas the units of the rest layers are artificial neurons that following the mathematical model described in the previous section (2.2.1). All the units of each layer are fully connected with all the units of the following layer (Figure 2.12.). The number of input and output units is determined by the structure of the input vector and the desired output. For example, to approximate the XOR logical operator, the input vector would be comprised of two input elements and the output target would be a single value; hence, the input layer of an MLP tasked to approximate the XOR operator would have two input units and one output neuron.

According to the Universal Approximation Theorem, MLPs are capable of approximating any measurable function to any degree of accuracy without any theoretical constraints for its success. When an MLP is unable to approximate a target function this is due to insufficient number of hidden units, *i.e.*, neurons in the hidden layers, or because the available data do not have deterministic relationship between the input and the target samples (Hornik et al., 1989). In addition, these statements are true for both one- and multi-dimensional problems.

Nevertheless, even if in theory an MLP with a single hidden layer of a finite number of neurons is capable of approximating any measurable function, in practice MLPs are usually designed with multiple hidden layers. This, however, introduces additional hyper-parameters (*e.g.*, number of hidden layers, number of neurons in each hidden layer *etc.*) that need to be tuned for constructing an efficient MLP for a specific problem. MLPs with multiple hidden layers may achieve a better generalization performance but are slower; on the other hand, an unnecessary increase in the number of hidden layers and their neurons may lead to overfitting. In general, ways to efficiently design an MLP for a specific task is still an open issue in the ML research and it can be regarded as a limitation of these models (Panchal, Ganatra, Kosta, & Panchal, 2011; Uzair & Jamil, 2020).

Table 2.3 XOR Truth Table

Input		Output
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

An example to demonstrate how an MLP manages to solve the XOR classification problem with a single hidden layer is provided. The truth table of the XOR gate is presented in Table 2.3. The XOR gate can be approximated by a three-layer MLP with two input units, a single hidden layer of two hidden neurons and an output layer of one neuron. An illustration of an MLP already trained to correctly approximate the XOR is illustrated in Figure 2.13. Each neuron of this MLP follows the mathematical model of Eq. (2.20) where the biases b_i, i = 1, 2, 3 and weights of each neuron are presented in Figure 2.13. For this example, as an activation function φ, the following step function was chosen:

$$\varphi(x) = \begin{cases} 1, x \geq 0.5 \\ 0, x < 0.5 \end{cases} \tag{2.23}$$

Once the MLP is well-defined and trained is capable of approximating the XOR logic gate and predicting the output of the XOR given a set of inputs. Let the vector $x = (0, 1)^T$ be the input of the MLP, *i.e.*, $x_1 = 0$ and $x_2 = 1$. According to Table 2.3 the output of the MLP given x should be 1. By performing the respective calculations, we have:

1st Hidden Neuron: $$N_1 = \varphi(-1.5 + 1 * 1 + 0 * 1 = -0.5) = 0 \tag{2.24}$$

2nd Hidden Neuron: $$N_2 = \varphi(-0.5 + 1 * 1 + 0 * 1 = 0.5) = 1 \tag{2.25}$$

Output Neuron: $$O = \varphi(-0.5 + N_1 * -1 + N_2 * 1 = 0.5) = 1 \tag{2.26}$$

As it can be observed in Eq. (2.26), the MLP that is used predicted correctly that the output of the XOR would be 1. By trying other combinations as input, it can be concluded that the respective network is indeed capable of correctly predicting the output of the XOR logic gate.

MLPs are arguably one of the most generalized forms of artificial neural networks, proven to be capable of learning any measurable function. There are some limitations, however. Such algorithms receive inputs in the form of vectors, and as a result multidimensional data, such as images, cannot be directly propagated to MLPs since they need to be vectorized. Vectorized images, *i.e.*, unraveled images in the form of a vector instead of an array, or other multi-

dimensional data increase dramatically the parameters of the first hidden layer of the network, and the approximation of these parameters is computationally demanding during the training.

2.2.3 Feature Extraction and Selection

Figure 2.14. Illustration of ML pipeline

The training of MLPs involves pairs of input and target data. By "showing" an MLP these input-target pair examples during training, enables the learning process during which the network fine tunes its weights and biases to provide accurate prediction regarding a classification or regression task. After its training the network is capable of providing the correct target results given a respective input. The training data that an MLP is capable of handling are in the form of N-dimensional vectors. Each vector that comprises the input data represents an observation, and the target output that corresponds to that observation represent the outcome that we want to approximate given the respective observation. Each observation is also called a feature vector. The elements of these feature vectors are observable quantities that are called features.

There are two main steps that are required for the efficient training of an ML algorithm, namely, the feature extraction and feature selection. Feature extraction enables the abstraction of information that subsequently leads to the reduction of its dimensionality. Feature selection aims at leveraging the most meaningful features that have been extracted. For an MLP to handle multidimensional data, e.g., images, certain features need to be extracted from each image. As it is discussed below (see section 2.2.4), by using entire images as input in an MLP, results to the vast increase to its parameters that makes its training computationally intensive and time consuming. For this reason, various methods have been proposed that can detect salient points, *i.e.*, single pixels or spatial regions in an image that are considered important. Once these salient points are detected, various features that encode information regarding the color, texture *etc.*, can be extracted and provided to an MLP to be trained and learn a given task. The feature extraction process leads to the construction of a dataset containing feature-target output pairs.

In most cases, a constructed dataset used for the training of an MLP contains noisy observations or observable quantities that are not correlated to the desired target output, *i.e.*, are irrelevant. The inclusion of such samples in the training process may lead to an unreliable predictive model that has low accuracy or is unstable. To cope with this problem, the datasets that are used for training an MLP usually require manual curation by the user to determine which observable quantities are useful or which observations are noisy to ultimately be excluded from the learning procedure. This process is called Feature Selection.

Feature selection enhances the training process by reducing the training time (by excluding feature vectors), increases the accuracy of the model by removing irrelevant or noisy data (Doraisamy, Golzari, Mohd, Sulaiman, & Udzir, 2008). Furthermore, by reducing the features incorporated in the decision-making process of the model a better understanding regarding which features affect the predictive outcome can be achieved (Arauzo-Azofra, Aznarte, & Benıtez, 2011). Algorithms that have been developed for feature selection can be organized in three categories, namely, filters, wrappers and embedded selectors.

Filter algorithms perform a process that aims at selecting the appropriate feature for a given task prior to the training of a classifier (Duch, Winiarski, Biesiada, & Kachel, 2003). Such algorithms evaluate the features and rank them according to their relevance. After the ranking the filter algorithms conclude to a set of superior features that can be used for training efficiently a classifier. Hence, filter algorithms are classifier invariant since they only examine the quality of the data and disregard the classification or regression algorithm that will be utilized.

Wrapper algorithms, in contrast to filter algorithms, are dependent on the classifier. These feature selection methods, consider a subset of the feature set and proceed to evaluate the performance of the ML algorithm using only this subset (Ron & George, 1997). This process is repeated for various subsets of the initial feature set. After many repetitions the optimum subset that leads to a superior performance of the ML algorithm is selected for training and inference. Wrapper algorithms are considered more reliable than filters, however, are more time consuming and computationally intensive since they require many trials to determine the best feature subset that enables a particular ML algorithm to solve a specific problem. In addition, the employment of a different ML algorithm leads to the selection of a different feature subset. This drawback is tackled by using some heuristic methods like genetic algorithms, greedy stepwise, best first, or random search. Nevertheless, wrappers become a prohibitive approach when the dataset at hand is high dimensional.

Embedded methods for feature selection aiming to determine the optimum features during the execution of the ML algorithm. Their name derives from the fact that they are a part of the ML algorithm itself either as its normal or extended functionality. Embedded feature selection algorithms include among others, decision trees and multinomial logistic regression and its variants (Cawley, Talbot, & Girolami, 2006; Sandri & Zuccolotto, 2006). Other embedded feature selection approaches incorporate weighting on the features based on regularization models that use loss function to simultaneously minimize the fitting error and the feature coefficients to be small or zero.

2.2.4 Limitations

As it was described in the previous sections, MLP is one of the most generalized universal approximators capable of learning any measurable function given a hidden layer with finite number of artificial neurons. Nevertheless, MLPs have its disadvantages, *e.g.*, MLPs can only process data in the form of vectors, has several hyperparameters, input data (features) need to be

engineered by humans *etc*. Some of these drawbacks that MLPs are posed to have been tackled with more recent and specialized models. This section summarizes these limitations in the following categories.

(1) Computational Complexity: The computational complexity of an MLP during the training is directly correlated with the number of connections between its layers. Since each neuron is connected to all the neurons of the next layer, the number of trainable parameters of an MLP model can greatly increase according to its architecture and the size of the input vector. The formula that can be used to estimate the parameters between two layers is the following:

$$p_k = n_K \cdot n_{K-1} + n_k \tag{2.27}$$

where p_k and n_K are the number of trainable parameters and neurons of the K^{th} MLP layer, respectively. For example, for a three-layer MLP with 100 input units, 256 hidden and 1 output units the total parameters can be computed as follows:

$$p_{total} = (256 \cdot 100 + 256) + (256 \cdot 1 + 1) = 26{,}113 \tag{2.28}$$

For a simple network like this, the trainable parameters according to Eq. (2.28) are 26,113. This number can dramatically increase with more complex MLPs that utilize more neurons and layers. Hence, MLP cannot be efficiently trained in scenarios where the input data consists of very large vectors and the complexity of the problem requires many neurons or additional layers.

(2) Tuning of Hyperparameters: By disregarding the choice of activation functions and training algorithms (see section 2.4) MLPs still have a lot of hyperparameters that need tuning. For example, even if with a single hidden layer an MLP can in theory approximate any function, the number of neurons that is required still needs to be determined. In practice, MLPs designed to be applied in various tasks usually utilize more than one layers where the number of neuros that each layer comprises still needs to be determined. It is obvious that the determination of the optimum number of layers and neurons it is not an easy task. Usually, with more than one hidden layer the accuracy of the MLP increases with a trade off in training time and inference (since the number of parameters increase, see Eq. (2.27)). However, if not the optimum number of neurons is found, the number of layers required to tackled successfully a particular problem may be difficult to determine (Uzair & Jamil, 2020).

(3) Limitations in Computer Vision: Modern CV requires the efficient processing of huge image and video datasets of high resolution. This makes the incorporation of MLPs in such tasks a tedious process since we can easily conclude that it would require a huge number of trainable parameters for training and using an MLP in that context. The "fully-connected" nature that characterizes the layers of MLPs makes them incapable of handling CV tasks easily just with the utilization of images as input. For example, the images of the MNIST dataset have a resolution of 32×32 that to be used by an MLP they initially need to be "flattened" to a vector form with a dimension of 1×1024. If we were to use a simple three-layer MLP with a hidden

layer that comprises 256 neurons and 9 output neurons (for classifying the numbers between 0-9 that are depicted in MNIST images), then the total parameters of that MLP would be 264,713. For a real-life task, if that three-layer MLP was to process a Full HD RGB image (1920×1080×3) for binary classification, *i.e.*, one output neuron, it would require to learn 1,592,525,313 parameters. That would require immense computational power for just training a simple MLP with modern CV data.

(4) Hand-Cafted Extraction and Selection: As it was described in section 2.2.3, an MLP to be trained properly demands curated pairs of input and target data. The inputs that are being used by an MLP to infer a prediction need to be correlated and well represented in the dataset for accurate results. The whole process of data curation is not an easy task, and the user needs to manually process them. As a result, this procedure is very time-consuming, and mistakes or inconsistencies can lead in unstable models with low accuracy. In the case of CV applications, due to the increased computational complexity that is introduced to an MLP by processing entire images, the use of algorithms that detect salient points from which hand-crafted features are extracted, is necessary, *e.g.*, abnormality detection in the gastrointestinal tract is benefited by color information whereas visual odometry by both color and lightness (George Dimas, Spyrou, Iakovidis, & Koulaouzidis, 2017a; Dimitris K Iakovidis & Koulaouzidis, 2014). Therefore, the user needs to conclude which salient point detection algorithms and feature extractors are suitable for each problem. The task of determining the most suitable salient detector and features per problem can be considered as an additional hyperparameter that need to be found and fine-tuned for a well-trained and efficient MLP.

(5) Black Box: Recently ML algorithms, have been imposed under regulations that have as requirements, safeguards and restrictions regarding ML and automatic decision-making in general. MLPs, and other DL models, are considered black boxes. As a result, after their training on a dataset, even if they are providing correct predictions, is completely unknown to users which features are considered meaningful. This makes such models unreliable since in high-stake domains like medicine, even if accuracy is of major importance, the rational of an inference is valuable. For example, if the sole purpose of an ML algorithm is either to make a prediction or to determine through learning if there is a correlation between the input and the output data, then the term “black box” is not considered a problem. If, however, the application focuses on extracting additional knowledge from a prediction process, *e.g.*, find new cues that lead to the determination of a malignancy in biomedical data, or the interaction of driving mechanisms, then the “black-box” nature of such a model limits its usability and its applicability in commercial applications.

2.3 Deep Learning

Conventional ML algorithms, like MLPs, have various limitations; a major one being their inability to directly use raw data, *e.g.*, entire images. On the contrary, the data that are utilized to train an MLP, need to be pre-processed manually to determine which features of these data are useful and should be considered. This procedure demands domain expertise, the engineering of features that need to be extracted from the data and the selection of subsets of the most useful ones. In summary, conventional ML algorithms require a manual process where human users design data representations that enhance enables the efficient training and learning of ML models.

These representations, however, can be learned; and for this purpose various methods have been proposed that are capable of learning representations given raw data (Bengio, Courville, & Vincent, 2013). These methods can be used by learning models to inherit the ability of utilizing raw data and extract representations that are learned during the training phase. Hence, the representations are automatically designed to make an ML model provide its best performance when trained on a particular set of data for tasks like classification, detection, or regression. More recent representation learning approaches that have dominated various domains including CV, are referred to as Deep Learning (DL). The definition of DL as it is provided by (LeCun et al., 2015) is the following:

"*Deep Learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned.*"

The purpose of these non-linear modules is to subsequently transform the input raw data into a representation that can be used by the machine to tackle a specific task. In the case of classification, for example, these representations, from the viewpoint of the machine, are better at segregating the various classes that define the classification problem; in the case of object detection, these representations are better at segregating the target entity that the machine is tasked to detect. Such processes that transform a set of data to a different representation are occurring constantly in our everyday life. For example, our eyes can be regarded as a deep learning module that through evolution learned to transform photons (raw data) to a representation that is leveraged by our brain to generate a visual perception of our surroundings. Another analogy that can be used to better grasp the utility of deep learning modules is that of a digital camera. A digital camera is a set of non-linear modules that, similar to the biological eye, transforms photons (raw data) to a digital representation of a scene that can be understood by a machine and be viewed by a human. The fundamental difference between traditional ML and DL models is illustrated in Figure 2.15.

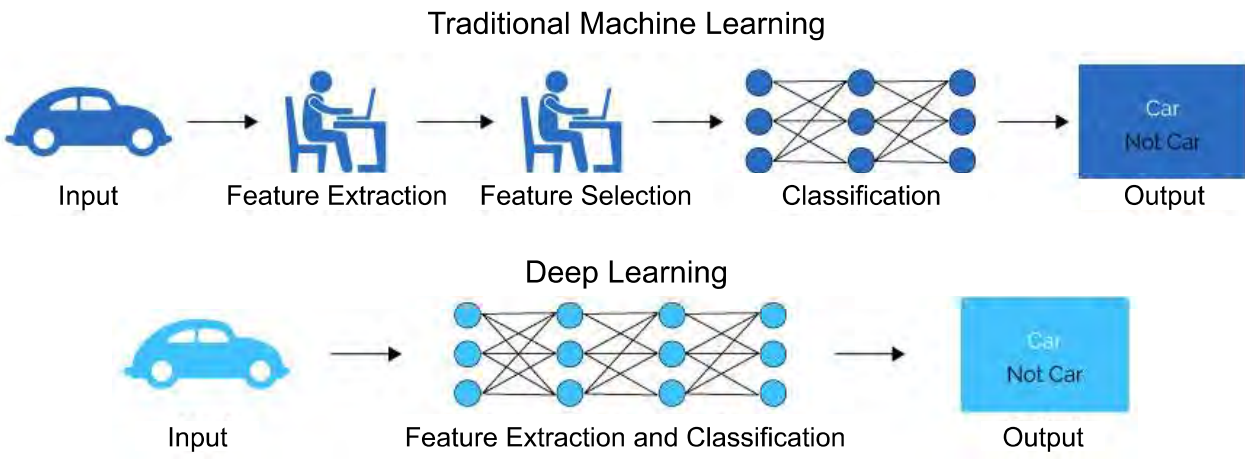


Figure 2.15. Illustration of the pipelines of traditional ML and DL

In the context of CV, a DL model that takes as input a colored image in the form of a tensor of pixel arrays, learns to subsequently transform it to various representations that up to a particular level of abstraction (*i.e.*, higher level) are similar to the features that are designed by humans. For example, the first layer (*i.e.*, first non-linear module) transforms the input tensor to a representation that encodes either the presence or absence of edges at different spatial regions of the image and orientations. The second layer, by leveraging the representation that is generated by the first one, learns to detect patterns that encodes various arrangements of edges. Similarly, the rest of the layers, regardless of the depth of the model (*i.e.*, the total number of layers) learn to leverage the representations of their previous layer to encode different features that enable the DL model to solve a particular problem. The key feature of DL models is that are capable of learning features, specifically designed for solving a particular problem, without any human intervention (LeCun et al., 2015).

Deep Learning models have dominated almost all the domains that deal with high dimensional data, making serious advancements towards the solution of problems that have been hard to tackle by using conventional ML. Deep Learning models, and specifically Convolutional Neural Networks have achieved record performance in CV tasks like image recognition, object detection, human gaze estimation and image generation. This dissertation, mainly focuses on understanding, evolving, and investigating the structure and capabilities of the CNN model that has been one of the most used and promising in DL.

2.3.1 Convolutional Neural Network

Similarly to MLPs, CNNs have been inspired by the design of the biological eye (Hubel & Wiesel, 1962). More specifically, a research team led by Fukushima managed to leverage the findings regarding the functionality of the cells in the visual cortex of cats, that was identified by Hubel & Wiesel, to create the predecessor of modern CNN, named Neocongnitron (Fukushima & Miyake, 1982). The first indication that CNNs have prospect to dominate the CV field was presented in

1989 by Yann LeCun and his team (LeCun et al., 1989). LeCun demonstrated that a CNN model trained using the backpropagation algorithm in a supervised manner is capable of classifying images of handwritten digits with satisfactory accuracy. However, the CNN revolutionized the Deep Learning and CV field in 2012 when the AlexNet deep CNN architecture was proposed (Krizhevsky, Sutskever, & Hinton, 2017).

The AlexNet architecture, to today's standards, is simple consisting of five convolutional and max-pooling layers followed by three Fully Connected (FC) layers. These three components, *i.e.*, convolutional, pooling and FC layers can be considered as the fundamental blocks of any CNN architecture designed to tackle classification tasks. Before diving into the structural details of different architectures we need to investigate the utility and usage of each of these components.

Convolutional Layers

A convolutional layer is the fundamental block of any CNN model. This layer, along with the CNNs in general, takes its name from a mathematical operation known as convolution. In the continuous space, a convolution is an integral that expresses the amount of overlap between a function g and a function f, as the function g is shifted over f, mathematically expressed as:

$$f(t) * g(t) = \int_0^{\tau} f(\tau) g(t-\tau) d\tau \tag{2.26}$$

In the discreet space, a convolution can be expressed as a summation process that expresses the amount of overlap between a g and f as the g is shifted over f :

$$s[n] * g[n] = \sum_{m=0}^{N} f[m] g[n-m] \tag{2.27}$$

In CV convolutions are widely used for applying various filters on images. The filtering of images is used for various tasks, from detecting specific features, *e.g.*, edges on images or smoothing the image by removing noise using the Gaussian filter. Firstly, let's see how the operation of convolution is used to apply a filter on a 2D discreet signal, *e.g.*, an image. Two dimensional signals, like images, are represented as $N \times M$ matrices. The filters that are usually applied on such signals are called kernels, and are also represented in the form of matrices of $n \times m$ dimensions

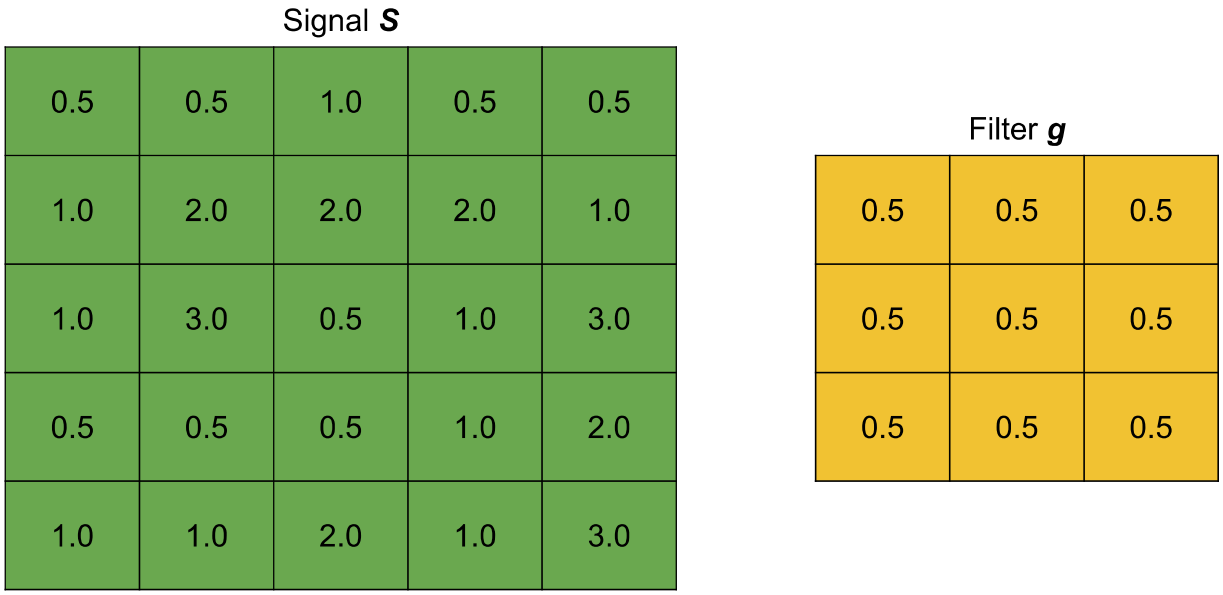


Figure 2.16. Visual representation of signal S and filter g.

where usually $N < n$ and $M < m$. In this section we will consider that all the 2D input and kernels have are squares. Let us consider, a 5×5 2D discreet signal S and a filter G with a kernel of 3×3 that are presented in detail in Figure 2.16.

The operation of convolution between two discreet 2D functions (or between a discreet signal and filter) is mathematically expressed as follows:

$$S[i,j] * G = \sum_{k=0}^{n} \sum_{l=0}^{m} S[i-k, j-l] G[k,l] \tag{2.28}$$

where $i = n, n + s, \ldots, N - n$ and $j = m, m + s, \ldots, M - m$ denotes the point that the filter is centered to perform the convolutional operation and s is the stride, *i.e.*, the step of the filter's shift on the 2D signal S. A visual representation of the result of the convolution considering signal S with the filter G at the point (1, 1) of S, given a stride of 1 is illustrated in Figure 2.17. As it can be seen the result of a such an operation has lower dimensions than the original signal. The red frame illustrates the neighborhood that the filter is applied in this first step. To define the size of the output we need to

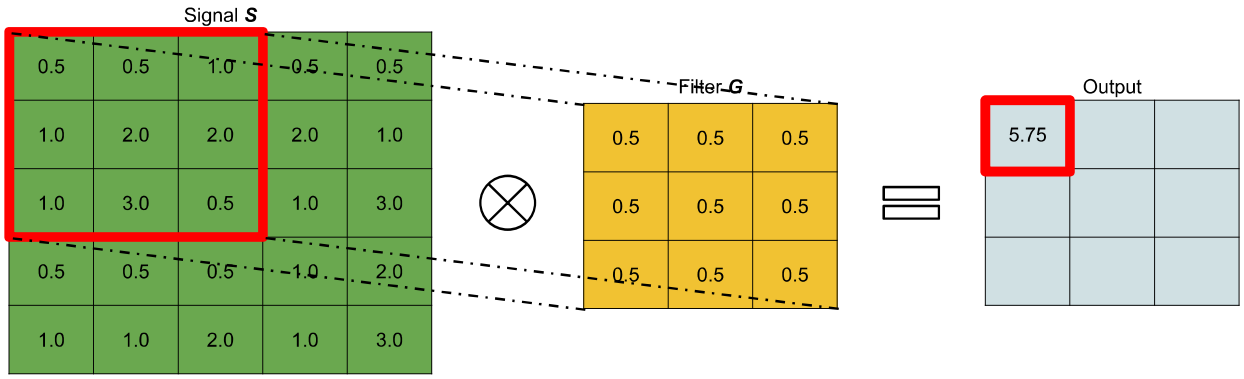


Figure 2.17. Visual representation of the result of convolution of *S* and *G* at the point (1, 1) of *S*.

consider the number of possible placements of the filter on the input signal. The possible placements are bounded in such way that the frame should be always inside the bounds of the signal. In our example, the first placement is at (1, 1) of *S*. Considering that the stride is equal to 1 the placements of the filter are the following: (1, 1), (1, 2), (1, 3) , (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3). Hence, the size of the output of a convolution between the *S* and *G* would be 3 × 3. A formula to determine the size of the output is expressed as follows:

$$o = (M - m) + (2 - s) \tag{2.29}$$

where o is the size of the dimensions of the output.

Figure 2.18. Whole convolutional progress along the input signal

There are cases however where the output of a convolutional operation needs to maintain the size of the input signal. In order to achieve this, a process is adopted that is called padding. Padding is the process where additional rows and columns are appended in the input signal for the output of

the convolution to have the same dimensions with the input. For example, in a 5×5 2D signal that is convolved with a 3×3 filter with a stride of 1, for the output to maintain the dimensions of the input, it required to additional rows and columns. In detail, is this example it requires 2 additional columns (one appended on the left and one on the right) and 2 additional rows (one appended on the top and one on the bottom). This process is illustrated in Figure 2.19.

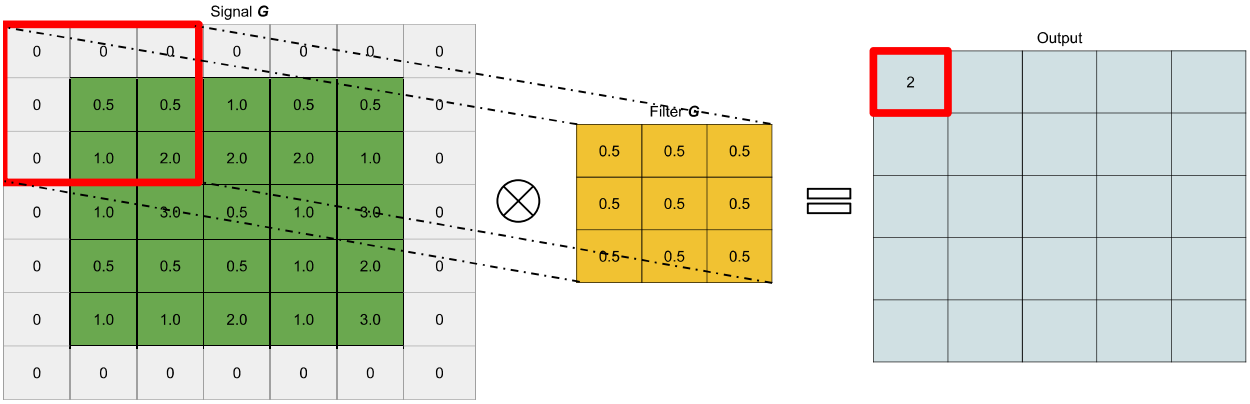


Figure 2.19. Illustration of the convolutional operation using padding.

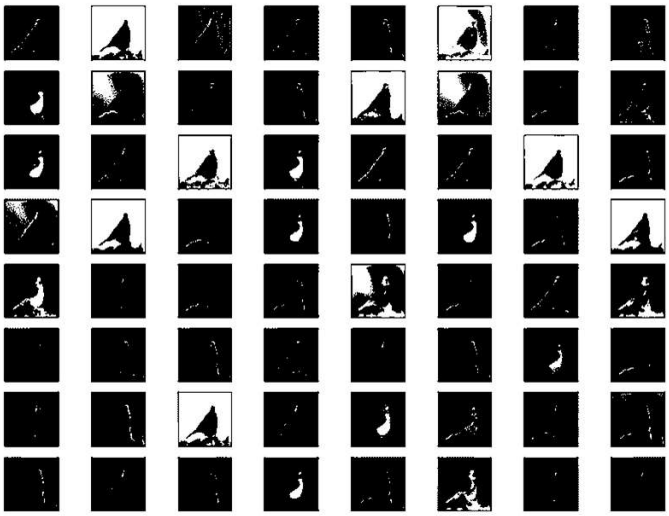


Figure 2.20. Example of feature maps that are estimated by a convolutional layer of a CNN model.

As it can be seen in Figure 2.19, the additional rows and columns are filled with 0 in this demonstration. This is called zero-padding which is the most common padding method; nevertheless, these additional positions can also be filled with average values from a border neighborhood of the input signal.

As it was described above, the convolution operation is the fundamental block of a convolutional layer. Each convolutional layer consists of multiple square filters (*i.e.*, kernels). The most common dimensions of these kernels are 3×3, 5×5, 7×7 and 1×1. These dimensions along with the number of filters that each convolutional layer utilizes are determined by the user that designs a CNN model. It should be noted that each individual kernel of a convolutional layer has a depth that is equal to the depth of the input that receives. For example, the first convolutional layer of the AlexNet filter CNN receives as input an RGB image and it is stated that it utilizes 96 kernels with dimensions of 11×11. Since each kernel is applied on an RGB image the first convolutional layer has 96 kernels with dimensions of 11×11×3.

Figure 2.21. Example of trained kernels of a convolutional layer of a CNN model.

The values of each kernel are the trainable parameters of the convolutional layer and are determined through training. We refer to the values of each kernel as weights. All the kernels comprise the weights of a convolutional layer. Each kernel has its own parameters which are not shared with other kernels in a convolutional layer. In this way, each kernel learns to describe a feature that may be contained in the input. The result of the convolutional operation of each kernel with the input is called feature map. Feature maps are 2D representations of the input that encode the presence or absence of a certain feature described by each kernel. Each convolutional layer outputs a number of feature maps equal to the number of kernels that it utilizes. For example, the first convolutional layer of AlexNet, since it has 96 kernels, it will output 96 feature maps. A visualization of feature maps and trained kernels of a convolutional layer are illustrated in Figure 2.20 and Figure 2.21, respectively. As it can be seen in the trained kernels, the training process guided them to resemble features that are similar to the edge detection filters with various orientations. The features that are represented by the kernels of each convolutional layer, however, are not engineered by a human (*i.e.*, hand-crafted), but are the product of a learning procedure where the deep learning model search on its own for the optimum representations that will benefit the convergence to a solution for a given task.

A convolutional layer, however, apart from the convolution of the kernels with the input it also introduces to the output a bias term and a non-linearity similarly to MLPs. Therefore, for an input of a 2D square input signal S with dimensions of $M \times M$ that has zero-padding so the output of the convolutional operation maintain same dimensions as S, a kernel of a convolutional layer w with a size $m \times m$ is applied on a position i, j of S as follows:

$$O[i,j] = \varphi\left(b + \sum_{k=0}^{m}\sum_{l=0}^{m} S[i-k, j-l]w[k,l]\right) \tag{2.11}$$

where O denotes the output feature map and $O[i, j]$ the position of the result of the left part of (2.11). The terms b and φ denote the bias term and activation function of the convolutional layer. In a CNN model architecture, the first convolutional layer receives as input the original 2D input signal, whereas the following convolutional layer receive as input the output of the previous layer.

Some noteworthy properties and specificities regarding convolutional layers in DL are summarized below:

- Each kernel is applied sequentially along the whole input; its weights remain unchanged during this process. Hence, each kernel can be regarded as a feature detector.

- The convolutional layer is translation invariant with respect to the input. This means, that if an object is shifted through an image it will not affect the response of the kernels since they search for specific features along the height and width of the input image.

- Convolutional layers are however affected by changes in orientation in the input signal. If a convolutional layer is not trained to detect changes in orientation, the feature detection process carried by the kernels will fail.

- Since each kernel considers the depth of the input we cannot dynamically change the input dimensions of convolutional layers. For example, if a convolutional layer is trained on RGB images it cannot process grayscale images since its each kernel will have 3 channels.

- The libraries that are available for the development of CNNs implement convolutions as cross-correlation, this however does not affect the results since in practice the same weights (but flipped) would be learned by the kernels even if were used in the context of conventional convolutions.

Pooling Layers

A common practice that satisfies both the dimensionality reduction problem, the reduction of computational intensity of CNNs, and the redundancy that may be introduced by overlapping convolutions when the stride is small, is the pooling technique. In CNNs, pooling is implemented in dedicated layers that usually are after a convolutional, or multiple convolutional layers. The pooling process is also inspired by biological mechanisms of the complex cells of the V1 visual area. Complex cells have been observed to combine local features that have been detected by simple cells of the visual cortex. The combination process is achieved by aggregating local features over a small spatial neighborhood. Similarly, the pooling layer has the functionality to aggregate local features that are located in a small spatial neighborhood of the estimated feature map of a convolutional layer. This process also enhance the translation invariance that describes CNNs (Scherer, Müller, & Behnke, 2010).

The most common approach that is used in CNNs to implement pooling is the aggregation of the values of a feature map using a sliding that selects non-overlapping neighborhoods of a feature map. There are different aggregation processes that are adopted in different CNN architectures; nevertheless, the most used are the following: *a)* max-pooling, *b)* average-pooling and *c)* global average- or max-pooling. Usually, all pooling approaches are used with a 2×2 sliding window (and a stride of 2 to avoid the aggregation of overlapping regions). Max-pooling, as it is indicated by the name, aggregates the values that are indicated by the sliding window of the pooling layer by

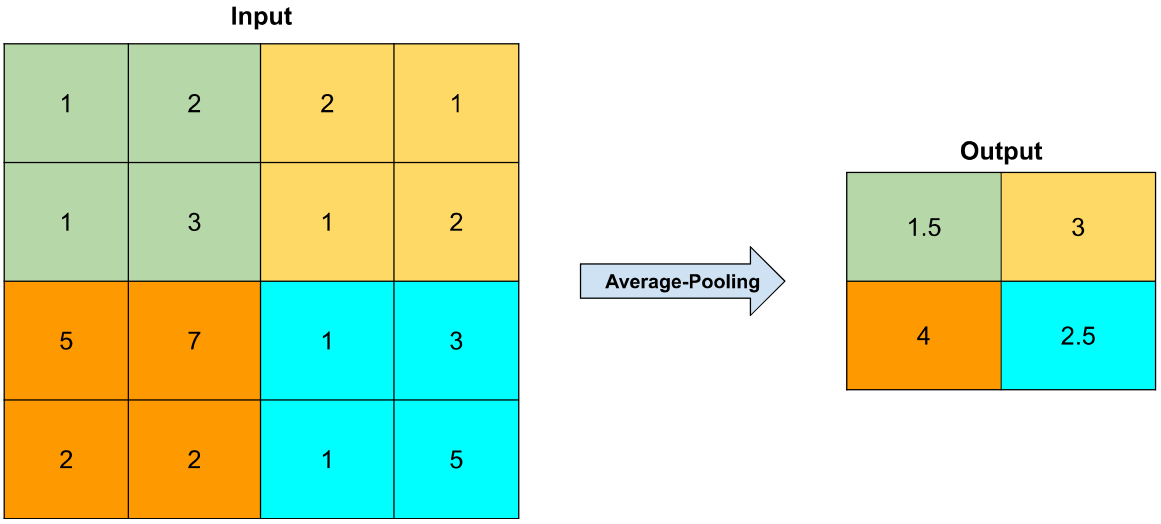


Figure 2.22. Example of the average-pooling layer functionality. Different colors indicate the non-overlapping spatial regions that are defined by the sliding window.

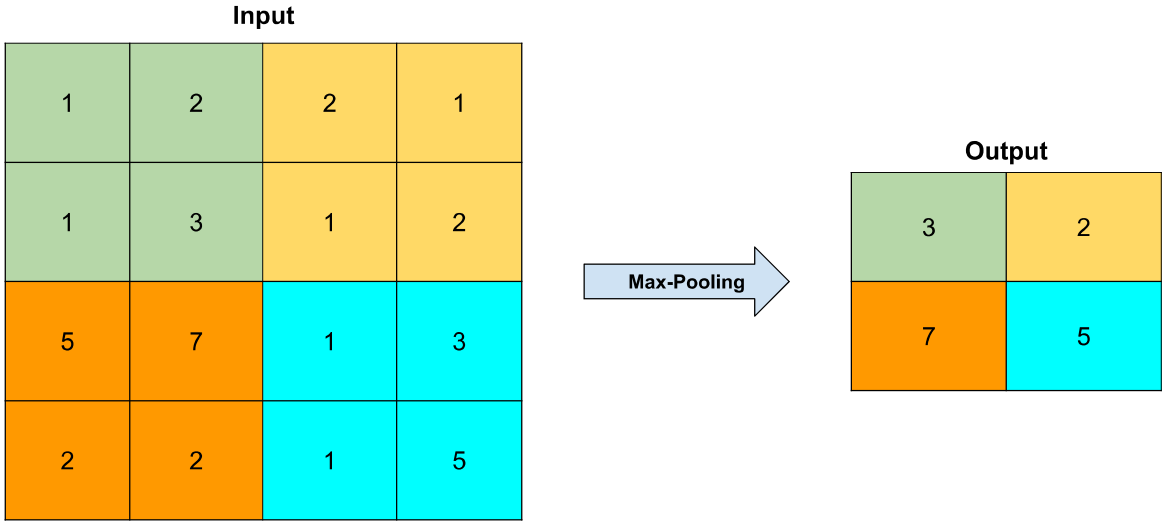


Figure 2.23. Example of the global-pooling layer functionality. Different colors indicate the non-overlapping spatial regions that are defined by the sliding window.

selecting the maximum value. The goal of max-pooling is to create a less complex representation, in terms of dimensionality, of the input feature maps that encode only the most dominant responses. Figure 2.23 illustrates an example of the max-pooling operation. The different colors indicate the different non-overlapping regions defined by the sliding window during the pooling process. On the other hand, average-pooling averages the values that are in the spatial region that is indicated by the sliding window (Figure 2.23).

Global-pooling takes a slightly different approach than simple max- and average-pooling. The output of the last convolutional layer is a set of feature maps each of which encodes the information regarding the presence and absence of specific low-level features. In the context of classification,

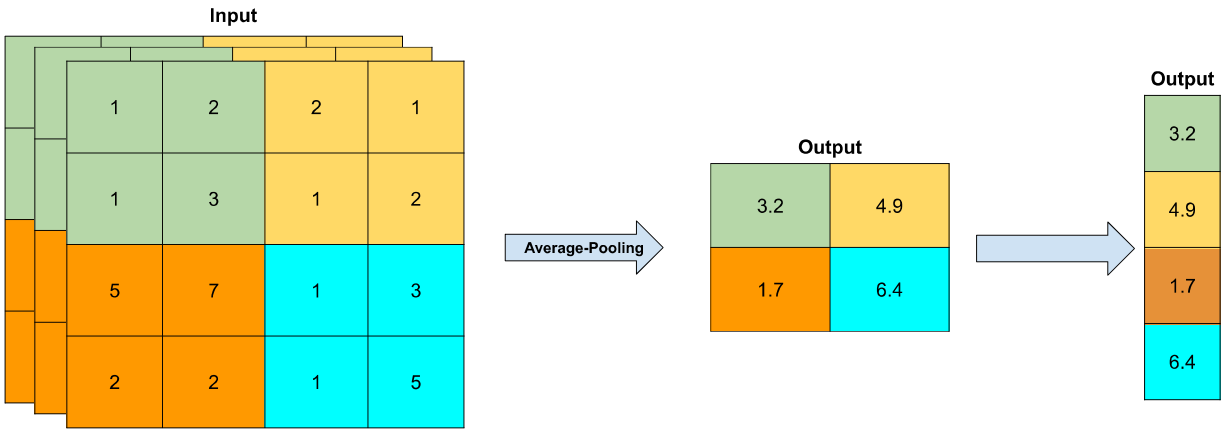


Figure 2.24. Example of the global-average-pooling layer functionality. Different colors indicate the non-overlapping spatial regions that are defined by the sliding window.

these feature maps need to be transformed in a vector form to be propagated to the following FC layers that are used to output the classification outcome. The transformation process of feature maps is used to make the CNN output compatible with traditional MLPs. Global pooling is used to remove this need and make CNNs capable of performing classification tasks without the need of FC layers. Global-average pooling is similar to the conventional average pooling with a slight twist: instead of limiting the sliding window on a spatial region along the height and width of a single feature map it considers this spatial region along the whole set of feature maps and averages all these values. As a result, global-average pooling extracts a single feature map that represent the average value of all the feature maps along the spatial region indicated by the sliding window.

An example of the global-average pooling layer is illustrated in Figure 2.24. As it can be noticed, regardless of the depth of the input, global-average pooling is capable of providing a single feature map as output that it can be easily propagated to the output layer that is responsible for estimating the classification or regression outcome. So, for an input of 4×4×3 the average and max pooling layers utilizing a 2×2 sliding window with a stride of 2 would provide and output of 2×2×3 whereas the global-average pooling using the same parameters would provide an output of 2×2×1. Global average pooling is more commonly used in modern CNN architectures like ResNet (He, Zhang, Ren, & Sun, 2016).

Pooling methods, however, even if they are widely used in almost all CNN architectures they are not considered as a best practice. The saying “data is holy” (Schmidhuber, 2007), that refers to the conservation of information at all stages of processing, contradicts the base principle of pooling approaches. These pooling methods, *i.e.*, max-, average- and global-average pooling contributes to the loss of information with one or the other way. For this reason recently in the literature pooling methods that aim at maintaining the information during pooling to enhance the performance of DL algorithms have been proposed (Diamantis & Iakovidis, 2020).

Fully Connected Layers

A CNN architecture is usually consisting of multiple convolutional layers that are followed by pooling layers. Then, after the final combination of convolutional and pooling layers, a set of Fully Connected (FC) layers follows. The purpose of the FC layers is to leverage the features estimated by the convolutional part of a CNN to perform a classification or regression task.

For the FC layers to be incorporated into a convolutional part of CNN architectures, the feature maps that are estimated by the final convolutional layer of a CNN need to be transformed to a vector form. After that, this vectorized form is propagated to the FC layers in a similar way with the features that are propagated in an MLP. Each FC layer consists of multiple artificial neurons that are fully connected to the neurons of the previous and next FC layer. The last FC layer that is designed according to the needs of the problem that needs to be tackled provides the prediction outcome.

A drawback of the FC layers is that they are computationally intensive during training and introduce an overhead of complexity to the CNN model that is proportional to the architecture of the FC part and the dimensionality of the feature maps that are estimated by final convolutional layer (see section 2.2.4). For this reason, in more recent architectures the global-average pooling layer is utilized.

2.3.2 Architectures of Convolutional Neural Networks

In recent years, the availability of large amounts of data and the innovation regarding the hardware required for their training and inference made CNNs a hot research topic. The advancements on the optimization of their training process, like fine tuning of their training parameters and especially the emergence of new CNN architectures achieving outstanding performance in image classification and recognition tasks have revolutionized the field of CV (Gu et al., 2018; Sinha, Verma, & Haidar, 2017). The term “model architecture” in the context of CNNs refers to the different combinations of various components that are utilized by CNNs, *e.g.*, convolutional, pooling and FC layers, in order to create a unified model that can be trained and solve a problem.

Current CNN architectures are designed based on convolutional blocks. Convolutional blocks are organized modules of multiple convolutional, pooling, and other layers. These blocks are repeated throughout the architecture with or without changes regarding the parameters of the convolutional layers, *e.g.*, size and number of kernels, strides, activation functions *etc*. The main focus regarding

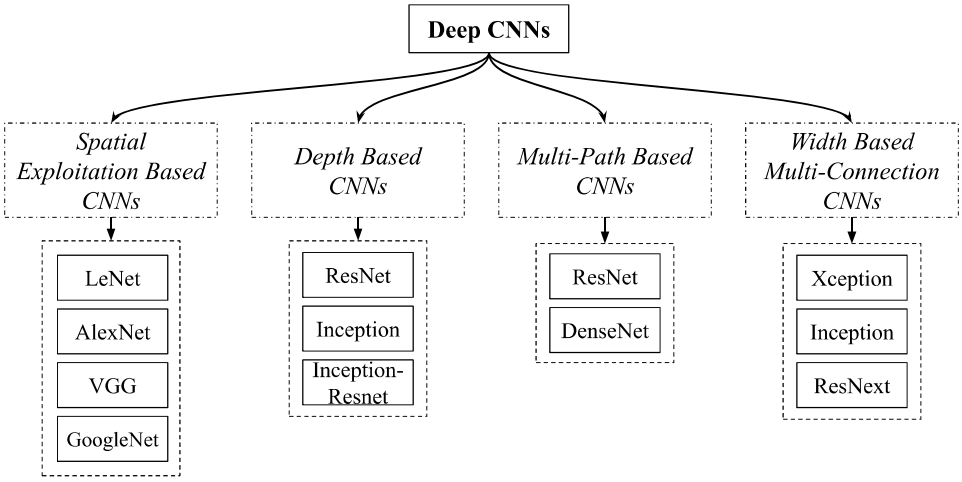


Figure 2.25. Summarization of the various CNN architectures along with the base logic of their development.

the investigation of new architectures revolves around the development of novel convolutional blocks that can be used in a modular way. This section summarizes the most important CNN and convolutional block architectures that have been proposed throughout the years. Figure 2.25 summarizes the various architectures that have been proposed in the literature throughout the years and provided game-changing performance in image classification and recognition tasks. In addition, Figure 2.25 presents the base principles that guided the development of the respective models.

LeNet CNN Architecture

LetNet CNN architecture was proposed in 1998 by Yan LeCun, Leon Bottou, Yoshua Bengio and Patrick Haffner (LeCun, Bottou, Bengio, & Haffner, 1998). LeNet is one of the earliest CNN models that has been used for recognizing handwritten characters. The architecture of this model is small since it consists of 2 convolutional layers 3 FC layers and 2 pooling layers. An illustration of LeNet can be seen in Figure 2.26.

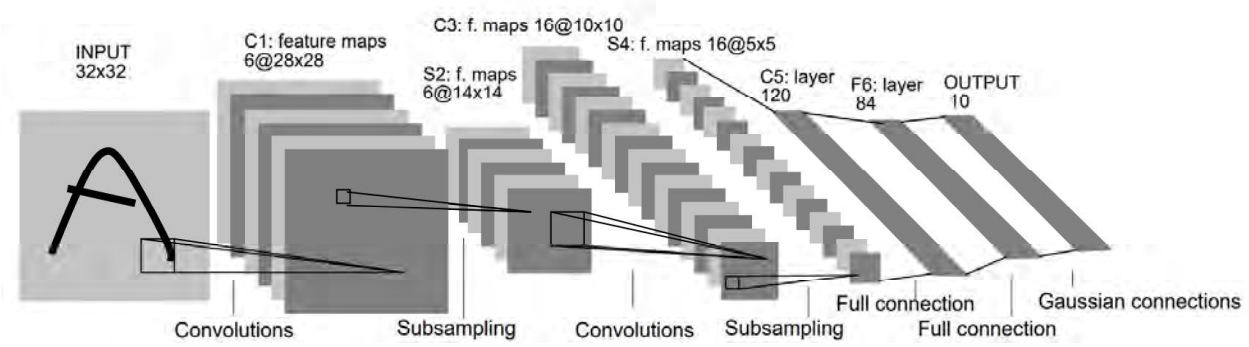


Figure 2.26. ©[1998] IEEE Reprinted with permission. Illustration of the LeNet architecture (LeCun et al., 1998)

This model was designed to classify grayscale images with dimensions of 32×32. The first convolutional layer utilized 6, 5×5 kernels that were performing convolutions with a stride of 1 on the input image. The second layer of this model was an average pooling layer that was applied on the feature maps estimated by the first layer with a sliding window of 2×2 and a stride of 2. The third and fourth layer of LeNet consists of a convolutional and an average pooling layer, respectively. This convolutional layer had 16 kernels with a dimension of 5×5 that are applied on the input with a stride of 1. The average pooling after the third layer had the same parameters as the previous pooling layer. The fifth, sixth and the output layer are FC layers with 128, 84 and 10 artificial neurons, respectively. The neurons of the final layer, *i.e.*, output layer, are decided according to the number of classes the model is tasked to categorize.

AlexNet CNN Architecture

The AlexNet CNN architecture was proposed in 2012 and it was trained for classifying RGB images of 1000 different classes of the ImageNet benchmark dataset (Krizhevsky et al., 2017). This model has a deeper architecture than AlexNet and consists of 5 convolutional layers, 3 FC layers and 3 pooling layers. The first convolutional layer comprises 96 kernels of 11×11 that are applied on the input with a stride of 4. The first convolutional layer is followed by a max-pooling layer that utilizes a 3×3 sliding window that shifts with a stride of 2. The third and fourth layers, similarly to the first two, are a convolutional followed by a max-pooling layer. This

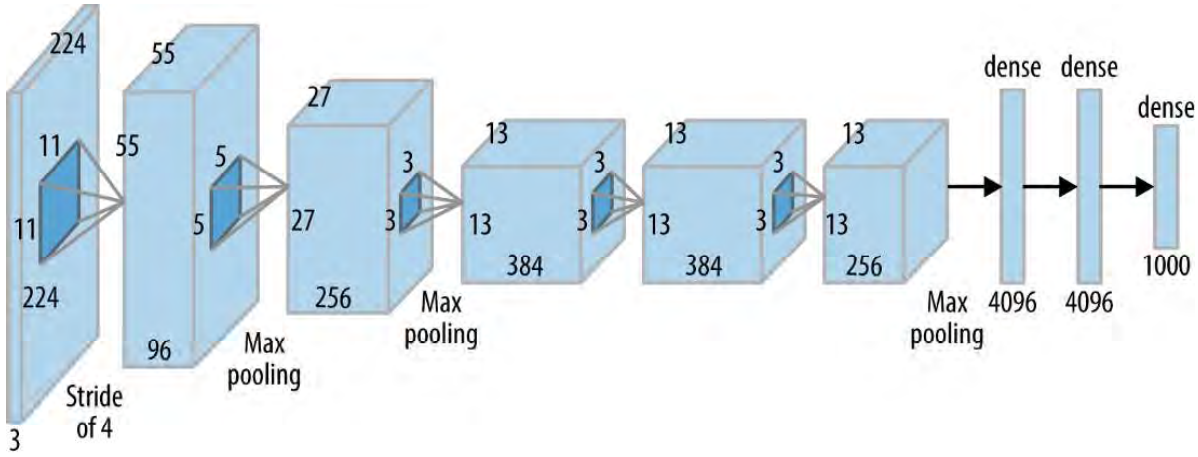


Figure 2.27. ©[2017] ACM Reprinted with permission. Illustration of the AlexNet CNN model architecture (Krizhevsky et al. 2017).

convolutional layer has 256 kernels with a size of 5×5 that are applied on the input of the previous layer with a stride of 1. In contrast to the first convolutional layer, this layer also uses padding on the input to manage the change in dimensions that is caused by the convolution. The max-pooling operation of the fourth layer had the same parameters as the previous max-pooling layer. The fifth, sixth and seventh layer are consecutive convolutional layers with the first two utilizing 384 kernels of 3×3 each, that are applied on the respective inputs with a stride of 1. The seventh layer has 256 kernels of 3×3 size that is also applied on the input with a stride of 1. These 3 layers utilized

padding in order the output of each layer to have the same dimensions with the input. After the convolution and the pooling layers AlexNet employs 2 FC layers and a final FC output layer. The first two FC layers have 4096 neurons whereas the output layer has 1000 neurons, same as the number of the classes that comprise the ImageNet dataset.

VGG CNN Architecture

The VGG CNN architecture has been proposed in 2014 introducing a very deep structural design for large-scale image recognition tasks (Simonyan & Zisserman, 2014). The VGG architecture employs small filters (kernels) that can encode the notion of up, down, right, left and center, *i.e.*, 3×3 size of kernels along with max-pooling and FC layers. VGG is structured in blocks of multiple convolutional layers followed by max-pooling operations. The most well-known and used VGG iteration is the VGG-16 version of the architecture. The number 16 indicates the number of convolutional and FC layers incorporated in this instantiation of the VGG architecture. In the original paper, the authors also proposed and tested two additional instances of the architecture, namely, the shallower VGG-11 and a deeper model VGG-19. Nevertheless, VGG-16 has been used in countless applications from the moment of its introduction until now.

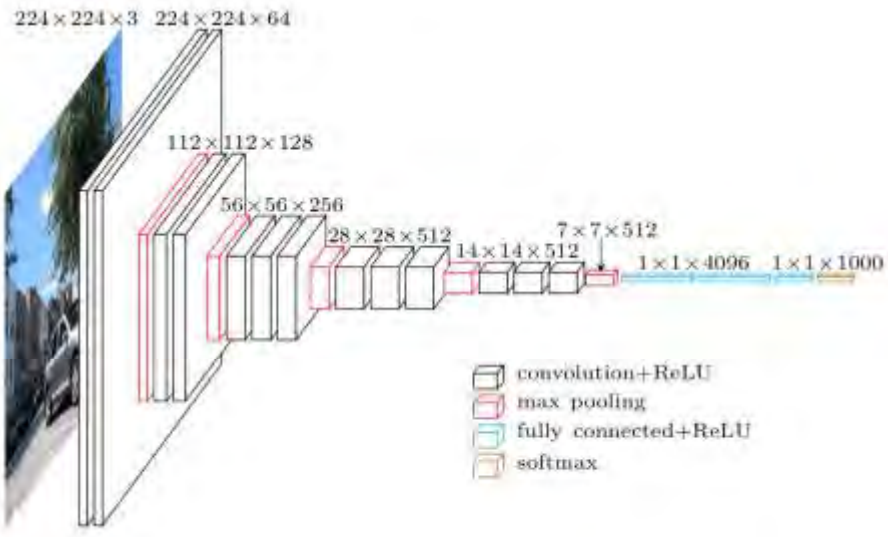


Figure 2.28. ©[2019] IOP Publishing. Illustration of the VGG-16 CNN architecture (Ming & Xu, 2019).

A visualization of the VGG-16 architecture is illustrated in Figure 2.28. VGG-16 consists of 13 convolutional and 5 max-pooling layers organized in 5 blocks followed by 3 FC layers organized in 5 different blocks. The 5 first blocks of VGG-16 have the role of the feature extractor whereas the 3 FC layers leverage these extracted features to perform the classification task. The final layer of each convolutional block is a max-pooling layer with a sliding window of 2×2 and a stride of 2; thus, only the convolutional layers will be described for each block. The first convolutional block of VGG-16 includes 2 convolutional layers with identical parameters, *i.e.*, 64 filters with a kernel size of 3×3 and a stride of 1. Like the first convolutional block, the second one has two

convolutional layers utilizing 128 filters with a kernel size of 3×3 and a stride of 1. The third convolutional block has 3 convolutional layers with 256 filters, a kernel size of 3×3 and a stride of 1. The fourth and fifth convolutional blocks are identical consisting of 3 convolutional layers each with 512 filters, a kernel size of 3×3 and a stride of 1. All the convolutional layers apply padding on their input to preserve the dimensions of the input after the convolutional operations. These convolutional blocks are followed by 2 FC and an output FC layer. The first 2 FC layers have 4096 neurons whereas the output layer has 1000 neurons, same as the number of classed the network is trained on. This model architecture managed to surpass the performance of AlexNet on the ImageNet dataset making VGG-16 the go-to model for tackling CV problems. The small size of kernels that VGG-16 uses are capable of extracting more "fine" features from images. As a result the feature maps that are estimated from each convolutional layer encode more information compared to the feature maps of AlexNet that uses bigger kernels.

GoogleNet CNN Architecture

The architectures of VGG and AlexNet are based on convolutional layers arranged in a sequential manner. A different approach has been adopted by the GoogleNet CNN model which introduced the Inception module (Szegedy et al., 2015). The authors that proposed the Inception module in the context of GoogleNet model, deviate from the conventional designing of sequential convolutions and proposed a parallel structure to their convolutional blocks. A visual representation of the Inception module is illustrated in Figure 2.29.

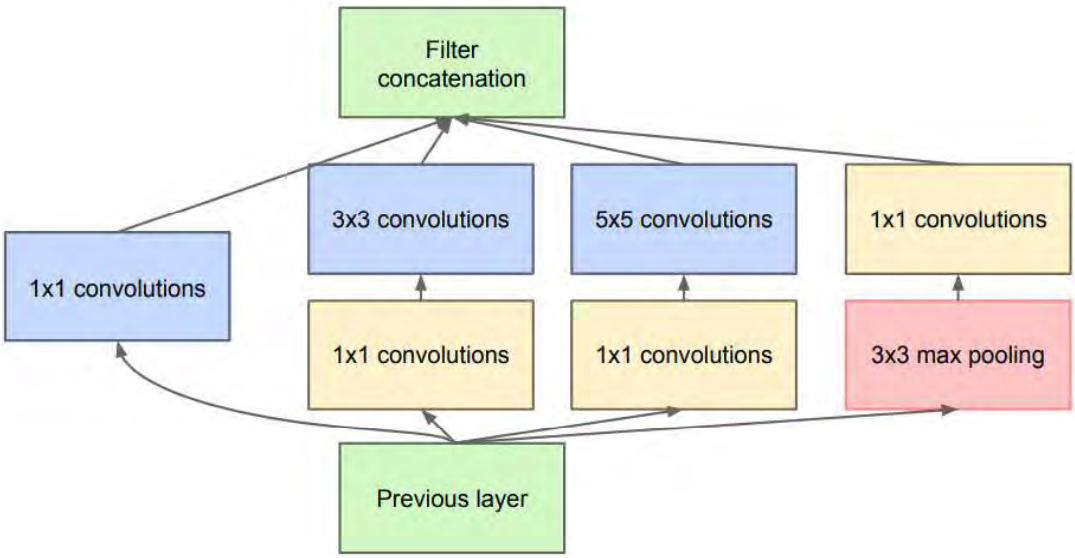


Figure 2.29. © [2015] IEEE. Illustration of the inception module design (Szegedy et al., 2015).

As it can be observed in Figure 2.29 the convolutional layers are structured in a parallel arrangement and utilize kernels of various sizes. The variations on size of the kernels aim at extracting features at different resolutions from the same input. Filters with kernels of smaller size can extract finer features whereas filters with kernels of bigger size extract coarser features. The

intention behind this design is the achievement of deeper CNN models capable of extracting more abstract features at different resolutions for achieving higher performance. The 1×1 convolutions are purposed to regulate the size of the input that is propagated to the more computationally intensive 3×3 and 5×5 convolutions. Furthermore, the output of all the convolutional operations is also concatenated with a pooled version of their input. Moreover, GoogleNet replaced FC layers after the convolutional part of the model with a global-average layer and an FC output layer. In this way the computational complexity of the model decreases. GoogleNet managed to outperform VGG models in terms of classification accuracy on the ImageNet benchmark dataset.

ResNet CNN Architecture

Another inspirational architecture that managed to outperform all the previous model that have been proposed and tested on the ImageNet benchmark dataset is called ResNet (He et al., 2016). The ResNet architecture was engineered to fulfill the need of deeper models that can be trained easily without facing the problem of vanishing gradient. The vanishing gradient problem emerges when an DL model is trained with a gradient-based optimization algorithm that utilizing backpropagation (Basodi, Ji, Zhang, & Pan, 2020; Hochreiter, 1998).

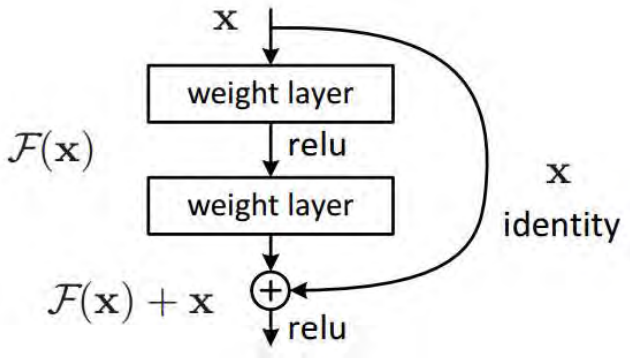


Figure 2.30. Illustration of a residual block (K. He et al., 2016a)

The problem of vanishing gradients usually occurs during the training of very deep CNN architectures, *e.g.*, in the VGG architecture, and in recurrent neural networks. A training process that incorporates these methods updates the weights of a DL model according to the partial derivative of the loss function with respect to the trainable parameters of the model. In the case of deep architectures, the weights close to the output layer can be updated without issues. Nevertheless, the partial derivative of the loss with respect to weights closer to the input layer can become vanishingly low, and as a result the weights of these layers closer to the input will either receive indifferent changes or will not be changed at all. As a result, in cases that vanishing gradient emerges the training of the model becomes more time consuming or, in the worst case, does not update its weights at all.

To cope with this problem and enable the construction of deeper CNN architecture, ResNet introduced the residual block (Figure 2.30). A residual block is constructed according to two main principles: the identity mapping and that multiple non-linear layers can approximate complicated functions. Let's use the notations of Figure 2.30 to better describe the residual block. In Figure 2.30 a convolutional layer (weight layer) followed by a ReLU non-linearity and another convolutional layer, given an input x are trained to approximate a function $F(x)$ which is similar to

the approach followed by conventional CNN architectures. The authors of ResNet go a step further and utilize the identity mapping, *i.e.*, the addition of x to $F(x)$ (*i.e.*, $F(x) + x$) to force these modules to learn a function that approximates a residual of the form $H(x) = F(x) + x \leftrightarrow F(x) = H(x) - x$. A restriction to residual blocks is that x and $F(x)$ must have the same dimensions for the successful summation.

This direction provides the error backpropagation two different paths during the derivation, one through $F(x)$ and another through the identity mapping, *i.e.*, through x. Hence, if the gradient vanishes during the error backpropagation through $F(x)$ the target weights can still be updated by computing the gradients through x. Similarly to the VGG architecture, the authors of ResNet proposed in their original paper multiple instantiations of ResNet of various degrees of depth (*i.e.*, number of convolutional layers). Additionally, in the same fashion as GoogleNet, ResNet utilize a global-average pooling layer followed by a single FC output instead of multiple FC layers to perform a classification or recognition task.

DenseNet CNN Architecture

Guided by the need for deeper and easy to train CNN models, in 2017 the Densely Connected Convolutional Neural Networks (DenseNet) have been proposed (G. Huang, Liu, Van Der Maaten, & Weinberger, 2017). This kind of architecture has been inspired by the FC neural networks that have been described in section 2.2.2. Previously, CNNs where designed in a feed forward fashion *i.e.*, a layer was propagating its output only to the following part of the model (a deviation of this trend can be found to ResNet were the input of a convolutional layer is added also to its output). DenseNet went a step further by introducing densely connected convolutional blocks. These convolutional blocks connect their layers to every other consecutive layer in a feed-forward fashion. In conventional CNN architectures a convolutional block with L layers would have L connections, *i.e.*, one between each layer and its subsequent layer, whereas dense blocks establish $\frac{L(L+1)}{2}$ direct connections. A visual representation of a network utilizing dense blocks is illustrated in Figure 2.31.

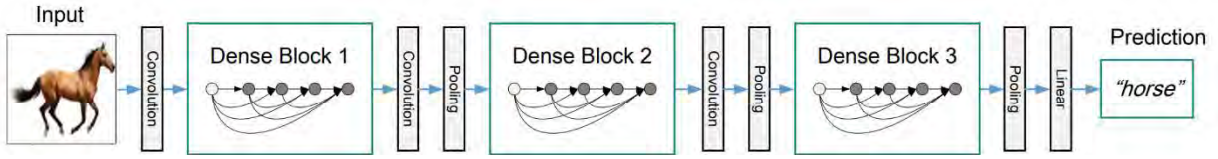


Figure 2.31. © [2015] IEEE. Illustration of a DenseNet CNN model (G. Huang et al., 2017).

In contrast to residual blocks where the information of a previous layer is added to the output of the next one, the connections in DenseNet are not formed in the basis of summation but with concatenation. In detail, the output of a layer is concatenated to the inputs of all the other layers. A restriction is that for the successful concatenation of feature maps their dimensions must be the same. Considering a dense convolutional block, the output of a layer H_l, where $l = 1, 2, 3 \ldots L$ is the index of a layer indicating its position in a dense block, is denoted as: $x_l = H([x_0, x_1, x_2, \ldots, x_{l-}$

$_{l}$]), where [·] denotes the operation of concatenation. This structure benefits the training and reduce the problem of vanishing gradients in a similar way with the residual connections. Since the output of layer *0* is present at the last layer *L* it provides different pathways to the backpropagation of the error for the computation of gradients that can effectively change the weights for an efficient training process.

Auto-Encoder CNN Architecture

The architectures that have been described previously are mainly used for image classification and regression problems that to be solved require the extraction of features from 2D signals, like images, and the output is a vector expressing the probability of an image to belong to a class, coordinates of bounding boxes, estimating the price of a house from an image etc. Such models employ a fully convolutional part, *i.e.*, a part consisting exclusively by convolutions connected with various ways to each other, followed by a single or multiple FC layers. However, there is another set of problems, that requires the extraction of features from an image that are utilized to generate a 2D map, e.g., class membership, on a pixel level. Such architectures usually receive as input an image and outputs a 2D map of the same, lower or higher dimensions to the input. These architectures are known as Auto-Encoders.

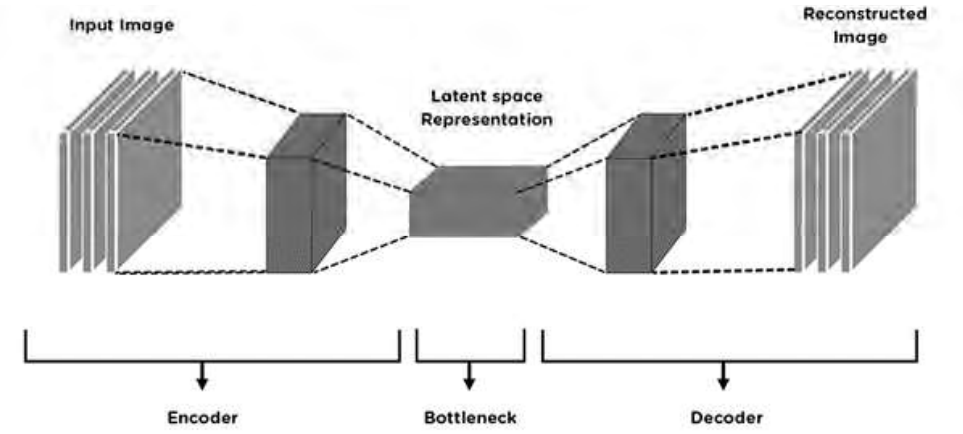


Figure 2.32. Illustration of the Auto-Encoder architecture

In the literature Auto-Encoders have been originally introduced in 1987 but with different terminology (Ballard, 1987; Schmidhuber, 2015). Convolutional Auto-Encoders consists of three parts: an encoder, a bottleneck and a decoder (Figure 2.32). The encoder has the same functionality with the convolutional part of a conventional CNN model used for image classification. An encoder can be constructed using multiple sequential convolutional blocks consisting of various components such as convolutional and pooling layers. The output of the encoding model is a representation of the input image, *i.e.*, an encoding of the input image, that is learned during the training and is shaped based on the task that the Auto-Encoder is trained to tackle. The output of the encoder is an unknown encoding of an image (unknown because it is learned by a black box) and it is also known as a latent representation of an image. Usually, the encoder has convolutional

blocks followed by pooling layers that are used to compress the input into a learned representation. The output section of the encoder is followed by the decoder part and is called bottleneck.

The purpose of the bottleneck is to suppress the information flow from the encoder to the decoder and is considered the most important part of the Auto-Encoder CNN model. Bottleneck distills the most important information to pass through to the decoder model. Another use of the bottleneck is that the compressed latent representation contributes to the avoidance of overfitting; the smaller the bottleneck (*i.e.*, the smaller the dimensions of the output of the encoder) the less chance for overfitting to emerge. However, if the bottleneck is too small, valuable information that would be necessary for the resolution of a task can be lost.

The final part of an Auto-Encoder is the decoder model. The decoder model receives as input the output of the encoder, *i.e.*, the latent representation, and outputs a 2D map relevant to the task that the Auto-Encoder CNN model is tasked to solve. This module of the Auto-Encoder, usually, has a symmetrical architecture to the encoder but in a reverse order and with the pooling layers replaced with up-sampling layers.

Originally the Auto-Encoder architecture was designed as an unsupervised method to encode and/or compress an input signal. This was achieved by training an Auto-Encoder to provide an output identical to the input. In the context of CV such architectures have been used for denoising, super resolution, gaze prediction *etc.* (Gatoula et al., 2021; Huaibo Huang, He, Sun, & Tan, 2017; K. Zhang, Zuo, Chen, Meng, & Zhang, 2017).

U-Net CNN Architecture

When Auto-Encoder CNN models are employed to different application, they maintain their main structure, *i.e.*, encoder, bottleneck, decoder, but they differentiate regarding other aspects such as the number of convolutional blocks they include in their structure. A noteworthy variation that revolutionized segmentation in biomedical images is that of the U-Net model (Ronneberger, Fischer, & Brox, 2015).

U-Net follows the main architectural principle of an Auto-Encoder but, in the same fashion with DenseNet or ResNet, changes the way its layers are connected. U-Net can be considered as an Auto-Encoder where each block of the encoder model is directly connected to its symmetric block of the decoder. An illustration to better grasp the structure and how the connections between the encoder and the decoder model are established can be seen in Figure 2.33. The connections between the encoder and the decoder are established by concatenating the output of the convolutional blocks of the encoder to the respective blocks of the decoder. The idea behind these connections is that the operation of up-sampling that is performed by the decoder is a sparse operation; thus, the information that is provided by the encoder introduces to the up-sampling process good priors for enhancing the performance of the network. In addition, in contrast to tasks like classification where the only information that is utilized is the output of the convolutional network, segmentation requires additional information regarding lower features to better determine

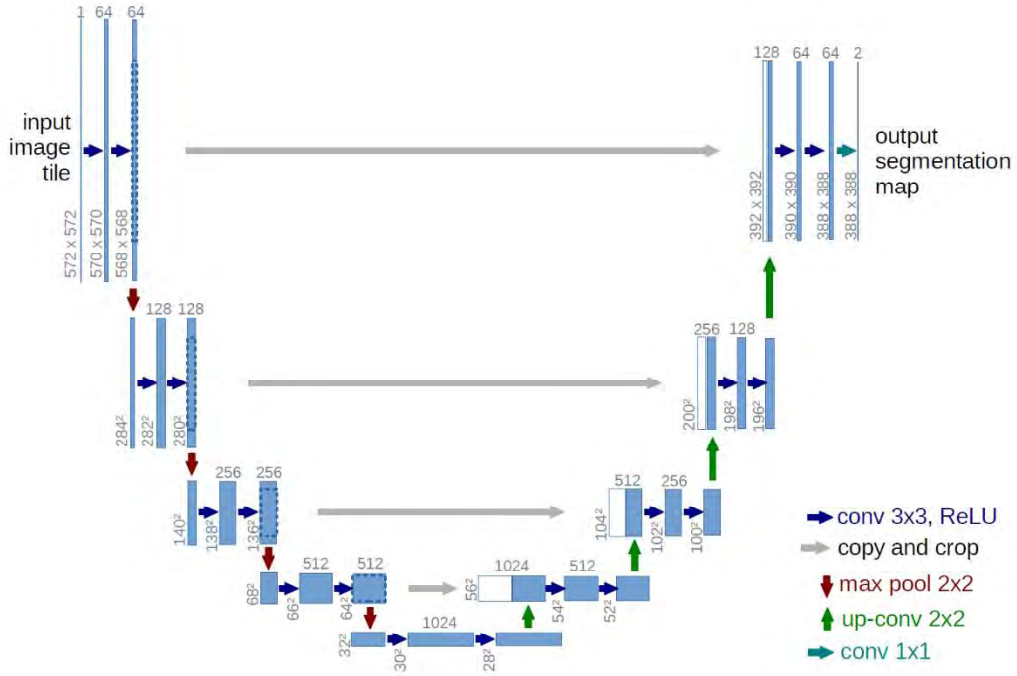


Figure 2.33. © [2015] Springer Nature. Illustration of U-Net architecture (Ronneberger et al., 2015).

boundary cues. Since its original introduction, many variations of the U-Net architecture have been proposed in the literature with changes that further enhance the performance (Huimin Huang et al., 2020; Z. Zhou, Rahman Siddiquee, Tajbakhsh, & Liang, 2018).

Ensemble CNN Architecture

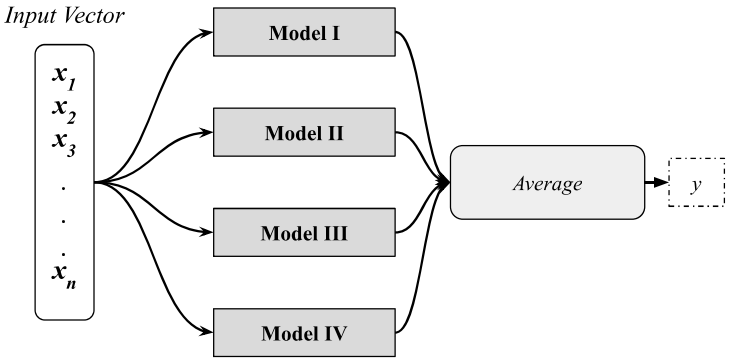


Figure 2.34. Illustration of an ensemble DL model.

Until now we have discussed architectures that used for building single models that are trained end-to-end and they deviate only regarding the structure of the convolutional blocks that use as building blocks. Another approach is to use multiple CNN models, of the same or different architecture, structured in parallel to form an ensemble of CNNs for performing classification tasks, usually through a voting process (Amin-Naji, Aghagolzadeh, & Ezoji, 2019; K. Liu, Zhang, & Pan, 2016). Often the models that are used in ensemble settings are trained separately on the same data and then their consensus is considered for deciding the outcome of a particular task. In other cases, the different models that

form a CNN ensemble are trained in a unified way by backpropagating the total error of the consensus.

A benefit of ensemble architectures of CNN models is that instead of very deep architectures, many shallow models can be used for constructing a unified network for performing CV tasks. With shallow architectures, the problem of vanishing gradients is reduced, and the training becomes a less complicated process. Furthermore, ensembles of networks enable the use of multimodal data. This can be achieved by stacking in parallel CNNs and MLPs. Thus, tabular data and images can be used as inputs at the same time in a model that can process both kinds of information at once.

2.4 Training The Learning Machines

Convolutional neural networks, and ML models in general, are data-driven representation learning methods. To learn these representations, such models need to be trained using input-output pairs, *e.g.*, images along with their respective class membership. During training, an ML model learns to map these inputs to the desirable outputs by examining the mapping error at each training step and trying to optimize its parameters to provide the correct result.

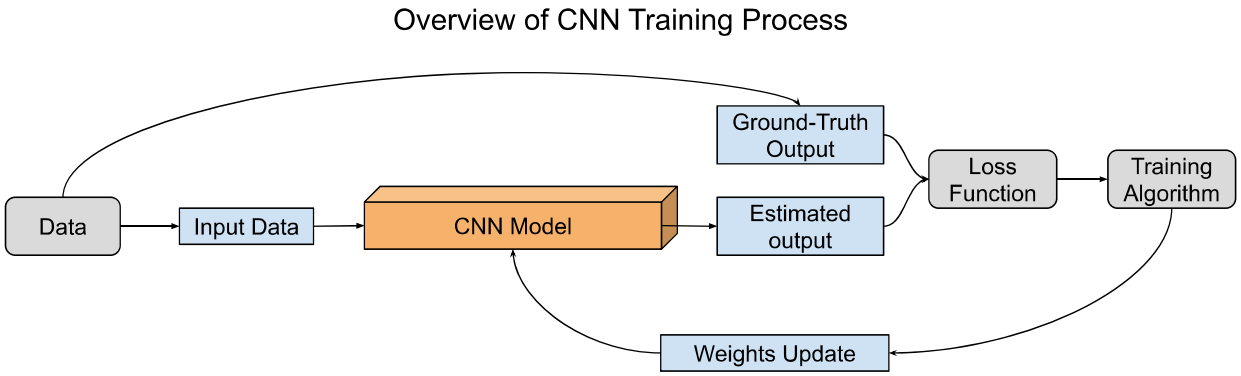


Figure 2.35. Overview of the training process of CNNs

The training of DL model is based on four pillars: *a*) data, *b*) loss function, *c*) the training algorithm and *d*) the learning paradigm. Data are of major importance for the learning process of CNNs. Deep CNN models require big amounts of data to learn tasks such as classification and recognition. The quality of these data is also a big factor to the training of CNN. Data annotated correctly can accelerate the convergence of CNN to solution, whereas bad quality annotations deteriorate the performance of a model. The loss function is a function of the form $L(y, \tilde{y})$ where y is the ground truth output, *i.e.*, the output that a network should provide, whereas $\tilde{y}$ is the output that the network estimates. Loss functions are used to assess the approximation capabilities of a model during training by estimating the error between the ground truth and estimated output. Loss functions are designed according to the problem at hand, i.e., binary classification problems require the use of a different loss function than multiclass classification problems. A training algorithm is responsible

for monitoring the estimated error of a model and utilizing the loss function to update the weights of the network through a process known as backpropagation. The learning paradigm describes the method that the data are used during training. The decision regarding which learning paradigm will be used at each case is mainly determined by the data that are available for solving a particular problem. An overview of the basic training pipeline of CNN models (and ML in general) is presented in Figure 2.35. In this section we will investigate the basics of loss functions and training algorithms.

2.4.1 Loss Functions

In the essence, a DL model is a function approximator algorithm which is trained to estimate the output of a function given a set of specific inputs. A DL model, such as CNNs can be formally denoted as follows:

$$\tilde{y} = f(x; \theta) \tag{2.31}$$

where $f(\cdot\,;\,\cdot)$ denotes the CNN model, x is an input to the model, θ is the parameterization of f and $\tilde{y}$ is the estimated output of $f(\cdot\,;\theta)$ given an input x. Loss functions are of the form $\mathscr{L}(\cdot,\cdot)$, they receive two inputs, the approximated output provided a DL model, and the ground truth output that f would output under ideal conditions (*i.e.*, if it was perfectly parameterized to model the target function). To incorporate a function $\mathscr{L}$ to the training process of a neural network it is required to be differentiable with respect to all the parameters θ of the model that we want to train. Loss functions are designed differently for each task. For example, the performance of a model when being trained to solve a binary classification problem is usually assessed using a loss function known as Binary Cross-Entropy (BCE) (Ruby & Yendapalli, 2020). On the other hand, multiclass classification problems require the employment of a loss function known as Categorical Cross-Entropy (CCE) (Zhilu Zhang & Sabuncu, 2018). Other widely used loss functions are the Mean Squared Error (MSE), Absolute Error (AE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE) and their formulation is presented below:

$$BCE(y, \tilde{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\tilde{y}_i) + (1 - y_i)\log(1 - \tilde{y}_i) \tag{2.32}$$

$$CCE(y, \tilde{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log\left(\tilde{y}_{ij}\right) \tag{2.33}$$

$$MSE(y, \tilde{y}) = \frac{1}{N}\sum_{i=1}^{N} (y_i - \tilde{y}_i)^2 \tag{2.34}$$

$$AE(y, \tilde{y}) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \tilde{y}_i| \tag{2.35}$$

$$MAPE(y, \tilde{y}) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \tilde{y}_i}{y_i}\right| \tag{2.36}$$

$$RMSE(y, \tilde{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2} \tag{2.37}$$

With N the number of training samples is denoted, y denotes ground truth value and in Eq. (2.33) C denotes the total number of classes incorporated in the learning process. Apart from the assessment of the error that occurs during training by the approximator, the loss function is also used during the back-propagation. During each training step, the backpropagation of the weights is achieved by deriving the loss-function with respect to the weights that we want to update. The loss functions that were referred earlier are some of the fundamental ones that are commonly used in the training of MLPs and CNNs. There are also many more loss function that have been developed during the years that are used in either in combination or standalone to enhance the performance of the models. Some game-changing loss functions that have been proposed in the literature are the adversarial, focal and perceptual losses (Goodfellow et al., 2020; Johnson, Alahi, & Fei-Fei, 2016; Lin, Goyal, Girshick, He, & Dollár, 2017).

2.4.2 Gradient Descent

A successful training of a CNN or MLP model is signaled when given a set of inputs the model provides outputs $\tilde{y}$ that minimize the responses of a loss function $\mathscr{L}$. For this to happen during the training the parameters of the model needs to be updated for the optimization of its performance. The task of calculating the change in the parameters that is required in order to minimize the responses of the loss function relies to the optimizer algorithm. Many optimization algorithms have been introduced for the efficient training of neural networks, however, the majority them are based on the iconic Gradient Descent (GD) algorithm. The gradient descent algorithm and its variations are considered as the go-to approach for optimizing the parameters of neural network model (Ruder, 2016).

The GD algorithm provides a method that aims at minimizing a loss function $\mathscr{L}(y, f(\cdot\,;\theta))$ by optimizing the parameters θ of f towards the opposite direction of the gradient of the loss function with respect to the parameters of f, *i.e.*, $\nabla_\theta \mathscr{L}(y, f(\cdot\,;\theta))$. To better grasp the logic behind this process, we can visualize an exemplary surface of $\mathscr{L}(y, f(\cdot\,;\theta))$. The gradient descent algorithm actually finds a path along the slope of the surface (gradient of $\mathscr{L}$ with respect to θ) that leads us from a high point of this surface (high responses of the loss function, big approximation error), downhill (error minimization) to a valley (local or global minima). The path downhill towards an

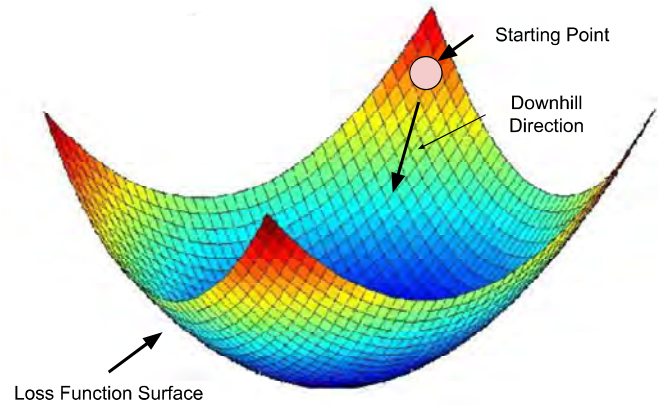


Figure 2.36. Visualization of the Gradient Descent process.

optimum parameterization of f that minimizes $\mathscr{L}$ is performed with a step η. The parameters η that is incorporated in the GD algorithm dictates how "*careful*" is the path discovery process towards an optimum solution. Big steps, *i.e.*, big values of η, may lead the parameterization process to lose an optimum solution whereas small steps, *i.e.*, big values of η, may lead to the increase of the training times since it will take more time to reach a possible solution.

A formal notation of the gradient descent step is the following:

$$\acute{\theta} = \theta - \eta \cdot \nabla_{\theta} \mathcal{L}(f(x;\theta), y) \tag{2.38}$$

where θ and $\acute{\theta}$ are the parameterization of f before and after the weight update performed by the gradient descent step, respectively. In the case of the vanilla version of gradient descent, the gradient regarding the whole dataset needs to be calculated to update the weights at each step (epoch). As a result, this version of gradient descent is time consuming and computationally intensive for the update of the weights at each step. For this purpose, different versions of gradient descent have been developed that can cope with this issue.

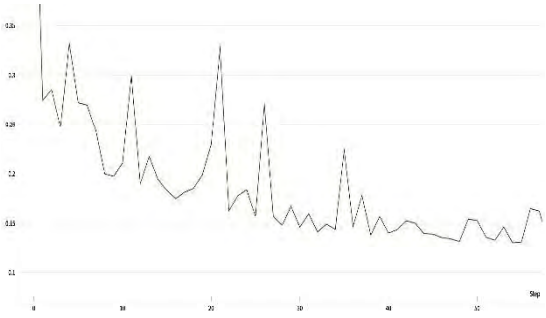


Figure 2.37. Example of SGD loss fluctuations

Stochastic GD (SGD) is a refined version of the original GD algorithm that tackles the calculation redundancy that has been noticed when training models on large datasets. SGD copes with this redundancy by updating the weights consecutively for each sample in the dataset at each epoch (instead one time for the whole dataset at each epoch). Hence, Eq. (2.38) can be rewritten to express SGD as:

$$\acute{\theta} = \theta - \eta \cdot \nabla_{\theta} \mathcal{L}(f(x_i;\theta), y_i) \tag{2.39}$$

where $i = 1, 2, 3, \ldots, N,$ denotes the index of training sample in the dataset. In short, with SGD the weights of neural network are updated as many times as the individual training samples in a dataset at each epoch whereas with GD the weights are updates once per epoch considering the whole dataset. A drawback of SGD is that introduces big fluctuations in the changes of the parameters (Figure 2.37), and as a result to the responses of $\mathscr{L}$, that can complicate convergence to the exact

minimum. By steadily and slowly decreasing the learning rate, however, this drawback can be overcome.

By combining the GD and SGD, we get a version of this training algorithm that is known as mini-batch GD. Mini-batch GD, splits the dataset in data batches, *i.e.*, groups of data of a certain size, and updates the weights of the model after the loss is calculated on each batch. To formally express the mini-batch version of GD Eq. (2.39) is rewritten as:

$$\acute{\theta} = \theta - \eta \cdot \nabla_{\theta} \mathcal{L}(f(x_{\{i+n\}}; \theta), y_{\{i+n\}}) \tag{2.40}$$

where $\{i + n\} = \{i, i + 1, i + 2, \ldots i + n\}$ denotes the indices of the training samples that are included in the current training batch, and $n = 1, 2, 3 \ldots N$ is the size of each training batch. The batch-sizes are usually a power of 2, *e.g.*, 8, 32, 64, 128 *etc*. With this approach the fluctuation that SGD introduces is reduced leading to a smoother training process with a more stable convergence to a minimum. Furthermore, the mini-batch approach can leverage the optimizations of the matrix operations that are implemented in recent DL libraries which highly accelerate the training process (Ruder, 2016). Nowadays, when we refer to SGD, we imply the mini-batch approach to GD which is the algorithm of choice for the training of neural networks.

New training algorithms are constantly introduced in the literature either in the form of refinements to existing ones, like the incorporation of momentum to SGD or the Nesterov accelerated gradient (Nesterov, 1983; Sutton, 1986), or by re-approaching SGD with novel methods. More recent alternatives to SGD are the *Adagrad, Adadelta, Adam, Adamax* and *Nadam* (Dozat, 2016; Duchi, Hazan, & Singer, 2011; Kingma & Ba, 2014; Zeiler, 2012). These new optimizers provide a more efficient way for the computation of gradients and the alternation of weights towards increased training stability and convergence to an optimum solution.

2.4.3 Backpropagation

It is often common to confuse GD with the backpropagation algorithm, and vice versa, as if it is the same thing. GD is the algorithm that provides a way to descent through the gradient of the loss function with respect to the network parameterization, *i.e.*, to optimize the weights of the network with the goal of minimizing of the loss function and achieving convergence of the model towards a solution. On the other hand, back propagation is the process through which the gradients of the loss function with respect to the parameters of the model are calculated throughout the network in a backward fashion starting from the output layer all the way to the input. Thus, backpropagation is a process that is applied prior to the GD. Once the gradients are calculated by the backpropagation algorithm, GD, or any other alternative, can be applied to update the weights of the network.

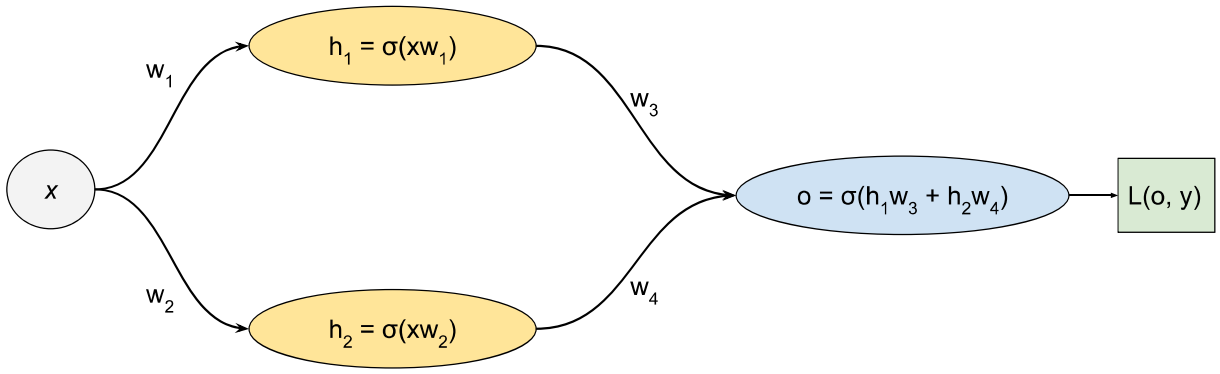


Figure 2.38. Illustration of a simple MLP.

According to Rojas, the backpropagation, applied on a simple MLP of three layers (one input, one hidden and one output layer), can be summarized in four simple steps (Rojas, 2013):

1) Feed-forward computation.

2) Backpropagation of the error to the output layer (*i.e.*, computation of gradients of the loss function with respect to the weights of the output layer).

3) Back propagation to the hidden layer (*i.e.*, computation of gradients of the loss function with respect to the weights of the hidden layer).

4) Update of the weights of the output and hidden layer (*i.e.*, exploiting the computed gradients to apply an algorithm like GD).

Now that we summarized the core of the algorithm let's elaborate on a bit more on these four steps:

Initially, we need to provide the model that we need to optimize with some training input samples to get an estimation that approximates our objective. Then, this estimation is used as input by a loss function along with a ground truth, *i.e.*, the real value that the model was tasked to estimate, to assess the error of the estimation. The input to the loss function is not just a scalar, or a vector of the estimate but the whole network.

After the assessment of the error, the backpropagation calculates the gradients of the loss function with respect to all the parameters of the network that need to be optimized. These gradients translate to how much each weight in its current state affects the error calculated by the loss function.

The calculated gradients are then used by an optimization algorithm to update all the trainable parameters of the network. The parameters are changed in order to minimize the loss function. For

this purpose, backpropagation uses a fundamental tool of calculus, the chain rule. In calculus, chain rule is a tool that expresses the derivative of the composition of two differentiable functions.

To better understand how backpropagation works we will need an example. Let $f(x; \theta)$ be a simple neural network with an input layer that receives a single input x, a single output layer and a hidden layer consisting of a single and two neurons, respectively. An illustration of this network can be seen in Figure 2.38, with $\theta = \{w_1, w_2, w_3, w_4\}$, $w_i \in \mathbb{R}$ and $\sigma(\cdot)$ denoting the activation function that is utilized by each neuron. For simplicity, the biases have not been considered in the parametrization of this network. Given a loss function $\mathscr{L}$ the assessment of the estimation error can be calculated as $\mathscr{L}(y, f(x; \theta))$ where it becomes apparent that $\mathscr{L}$ is differentiable with respect to the parameters of model f. By assuming that each layer is a function, the model f is actual a composite function, $f(x; \theta) = o(h(x; w_1, w_2); w_3, w_4) = o(h_1(x; w_1), h_2(x; w_2); w_3, w_4)$. Hence, the estimation of the loss can be expressed as:

$$\mathcal{L}(y, f(x;\ \theta))\ =\ \mathcal{L}(y, o(h_1(x; w_1), h_2(x; w_2); w_3, w_4)) \tag{2.31}$$

To estimate the gradients with respect to the parameters of the network f we need to estimate the partial derivatives of L with respect to all the weights of the network, *i.e.*, $\nabla_\theta L$. To achieve this, we utilize the chain rule as follows:

$$\frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_4} = \frac{\partial \mathcal{L}(y, o(\dots))}{\partial o(\dots)} * \frac{\partial o(\dots)}{\partial w_4} \tag{2.32}$$

$$\frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_3} = \frac{\partial \mathcal{L}(y, o(\dots))}{\partial o(\dots)} * \frac{\partial o(\dots)}{\partial w_3} \tag{2.33}$$

$$\frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_2} = \frac{\partial \mathcal{L}(y, o(\dots))}{\partial o(\dots)} * \frac{\partial o(\dots)}{\partial h_2(x; w_2)} * \frac{\partial h_2(x; w_2)}{\partial w_2} \tag{2.34}$$

$$\frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_1} = \frac{\partial \mathcal{L}(y, o(\dots))}{\partial o(\dots)} * \frac{\partial o(\dots)}{\partial h_1(x; w_1)} * \frac{\partial h_2(x; w_1)}{\partial w_1} \tag{2.35}$$

Once all the partial derivatives have been calculated using the backpropagation algorithm, they can be used iteratively to update the parameters of the network. For example, using the GD (Eq. 2.28) the parameters of the network are updated as follows:

$$\acute{w_4} = w_4 - \eta \cdot \frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_4} \tag{2.36}$$

$$\acute{w_3} = w_3 - \eta \cdot \frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_3} \tag{2.37}$$

$$\acute{w_2} = w_2 - \eta \cdot \frac{\partial \mathcal{L}(y, o(\dots))}{\partial w_2} \tag{2.38}$$

$$\acute{w_1} = w_1 - \eta \cdot \frac{\partial \mathcal{L}\left(y, o(\ldots)\right)}{\partial w_1} \tag{2.39}$$

This process is repeated throughout the training until the network converges to a solution with a low estimation error. Even if the process of backpropagation in this example has been demonstrated in the context of MLPs, the same process is applied on CNNs. The combination of backpropagation and the various approaches of GD is a very simple yet powerful training method.

2.4.4 Learning Paradigms

DL models like CNNs and MLPs are representation learning algorithms. In the previous chapter we defined and analyzed the process that takes place during the learning procedure of a model. It was mentioned many times throughout this manuscript, that the architecture and training of these algorithms is heavily inspired by biology, so are the methods of learning. There are many ways through which we can train a human to learn something. In all methods however the common ground is that the subjects that is learning something should be improved by observing the errors that makes. Similarly with DL models, there are different ways that the learning process can be approached, the observation of error and the optimization of the model through its errors however remains the same. Hence, a constant among all the learning approaches is the computation of gradients through backpropagation and the use of this information to update the weights towards a solution.

The choice of the learning approach that is used to train a DL model to solve a particular problem is decided considering the following factors: *a*) which component of the model needs improvement, *b*) the existence of the prior knowledge that exists in the model, *i.e.*, if it has been already trained to solve a different or similar problem and *c*) what kind of data (and feedback on the data) is available. Based on these three factors there are four main learning paradigms that exist, namely, supervised learning, self-supervised learning, unsupervised learning and transfer learning.

Supervised Learning

In the supervised learning paradigm, a DL model is trained by observing pairs of input and output data samples (Figure 2.39). Through the observation of these data, the model learns a representation of the input data that helps it to approximate a function that maps the input to the respective output. An example of such data in the context of CV are image-label pairs. The images are used as input to the model which is tasked to map the image to its respective label. Labels describe classes that are represented in an image. For example, if we want to train a model to

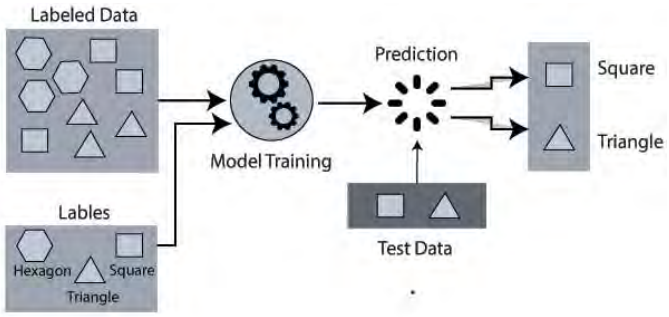


Figure 2.39. Example of Supervised Learning in ML

classify images that contain dogs or cats, the image labels would be "cat" or "dog" depending on whether either animal is depicted in an image.

The goal of the learning process is to optimize the model in such way that when it will be asked to classify an image that was not included in the training data to successfully predict its class. The agent learns a function that, when given a new image, predicts the appropriate label.

In a formal way the supervised learning describes the training process where given a training set of N examples of input-output pairs $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_N, y_N)\}$, where each data pair was generated by an unknown function $y_i = g(x_i)$, $i = 1, 2, 3, \ldots, N$, a ML or DL model f is tasked to approximate the true function g that generated the given data (Russell, 2010). These x_i are known as inputs and y_i as ground truth values, *i.e.*, the true value we are asking the model that we train to predict. The learning process should be in such way that the model is capable to approximate the output on the true function g when given an input that was not included in the training procedure.

Self-Supervised Learning

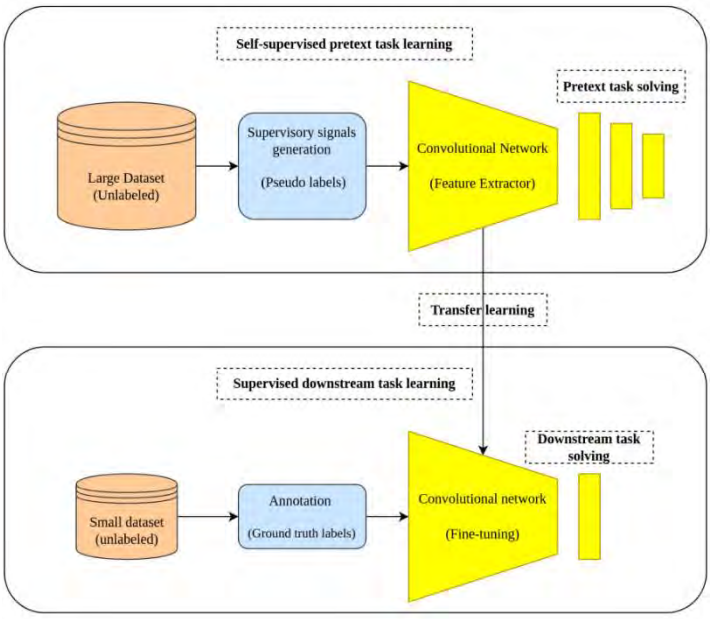


Figure 2.40. © [2022] Shurrab et al. Main pipeline of self-supervised learning (Shurrab & Duwairi, 2022) (DOI: 10.7717/peerjcs.1045/fig-1)

A more recent learning paradigm is the self-supervised learning (Figure 2.40) (Shurrab & Duwairi, 2022). In supervised learning, instead of using a set of already known input-output data pairs, self-supervision enables learning of semantic features with the generation of supervisory signals from a collection of unlabeled data, *i.e.*, only input data which the true function that provides the ground truth output is unknown, without the need for human annotation (Liang Chen et al., 2019). A model that is trained through the self-supervised paradigm, it can either directly deployed to provide predictions given certain inputs or its learned features can be used for subsequent tasks where the amount of the annotated data is limited.

The main component of self-supervised learning is the pretext task. During this task the model is trained in a supervised fashion using unlabeled data. For this process to be characterized as supervised the unlabeled data is required to be annotated. In contrast to supervised learning, the annotation occurs in an automated process through which the data are labeled in way that enables the model to learn useful representations.

The self-supervised learning paradigm gained ground in fields where the amount of unlabeled data is overwhelming. A domain that is characterized by the imbalanced quantity of labeled *vs.* unlabeled data is the biomedical domain. Hence, several methodologies have been proposed that demonstrate the effectiveness of the self-supervised learning in many applications like classification, detection, localization, and segmentation tasks (Liang Chen et al., 2019; Lu, Chen, Wang, Dillon, & Mahmood, 2019).

Unsupervised Learning

In contrast to self-supervised learning paradigm, unsupervised learning is a concept that has been proposed in the early days of ML (Barlow, 1989). Self-supervised learning, as described above, is often confused with unsupervised learning. Both unsupervised and self-supervised learning omit the need for manually annotated data, nevertheless, the self-supervised approach still requires some sort of annotation to utilize the available data. On the other hand, unsupervised learning does not require any form of annotation, only raw input data points (Ghahramani, 2003).

According to the unsupervised learning paradigm, a ML model is trained by simply utilizing inputs $x_1, x_2, \ldots, x_N$ without any supervised target output y or a reward system. This concept may be confusing since it is difficult to understand how a model can learn without a predetermined goal. The context that unsupervised learning is employed is that of finding patterns in unstructured data through some sort of training. A model that follows this learning paradigm, usually is trained to either reduce the dimensionality of the input signals or to learn some sort of representation of the input data that can organize them in clusters. An example of dimensionality reduction through unsupervised learning can be given with an Auto-Encoder model. An Auto-Encoder model can be trained in a way that given an input x to output an estimation $\tilde{y}$ that is the same as the input. In this way for its training only the input data x are required. Then, once its trained, the bottleneck of the Auto-Encoder is a latent representation of the input image with reduced dimensionality.

Transfer Learning

The transfer learning paradigm is used to further improve a ML model in a specific problem by transferring knowledge from another, related problem. This is usually the case when the data that are available for solving a problem are not enough for a model to approximate a solution. This approach is very similar to how humans are often tackling problems by combining information that have been acquired by past experiences. A simple example is the following: Imagine two people who are interested in learning how to play the piano. One person is yet to learn any musical instrument whereas the other person has extensive music knowledge through playing the guitar.

The person who has a background in musical studies, is more likely to learn faster how to play the piano compared to the person who does not know how to play any instrument. The accelerated learning that is very likely to occur to the person who knows how to play a musical instrument relies to fact that he can transfer previously learned music knowledge to the current learning task. In other words, a person is able to take information from a previously learned task and use it in a beneficial way to learn a related task (S. Pan & Yang, 2010).

The learning paradigm of transfer learning is employed when the data availability for solving a problem is limited. This is very common when we want to train a model for tackling a detection or recognition task in the biomedical domain. The ethics involved in this domain make the data usage and availability limited. Hence, to cope with this difficulty, a model can be initially trained on a big dataset that is available (*e.g.*, ImageNet), acquire the necessary knowledge regarding image processing and subsequently be fine-tuned on a smaller dataset. Many applications have successfully utilized transfer learning for the efficient training of ML models. These applications include text sentiment classification (C. Wang & Mahadevan, 2011), image classification (W. Li, Duan, Xu, & Tsang, 2013), human activity classification (W. Li et al., 2013), software defect classification (Nam & Kim, 2015), and multi-language text classification (J. T. Zhou, Tsang, Pan, & Tan, 2014).

2.5 Interpretability

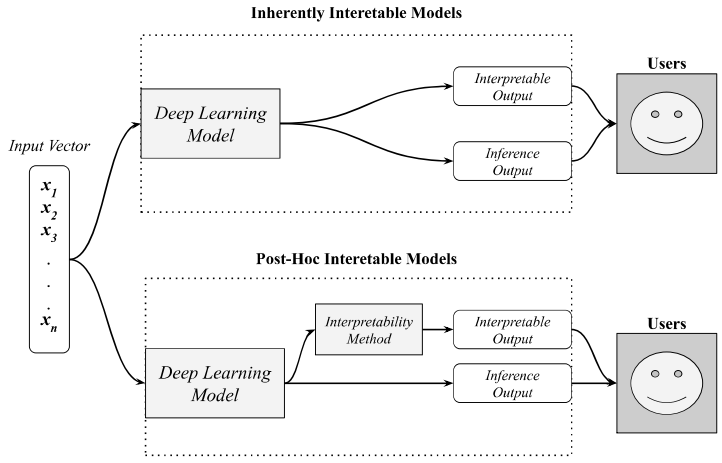


Figure 2.41. Visual example of interpretability pipeline

Recently the commercial applicability of ML algorithms has been regulated by legislation acts that aim at making the world 'fit for the digital age' with requirements, safeguards, and restrictions regarding ML and automatic decision-making in general (Selbst & Powles, 2018). Interpretability is a critical factor in the compatibility of ML models with these regulations. But how is the interpretability of machine learning models defined? According to recent literature, interpretability is a passive characteristic of a model that indicates how well a human understands the reason for its decision (Angelov, Soares, Jiang, Arnold, & Atkinson, 2021). As a result, providing interpretations of a model's decision-making process can reduce its opacity and earn users' trust, for example, by providing interpretations for risk-sensitive decisions in medicine. In real-world problems, ML models' discriminative capacity, as expressed by performance measures such as predictive accuracy, is regarded as an inadequate descriptor of their decisions (Rudin et al., 2022).

Various approaches to interpretability have been taken from a post hoc standpoint, i.e., methods that use a fitted black-box as input to determine the causality of its approximations (Murdoch, Singh, Kumbier, Abbasi-Asl, & Yu, 2019). Image perturbation methods used on the network including masking, substituting features with zero or random counterfactual instances, occlusion, conditional sampling, and so on. Such approaches seek to identify image regions that have an impact on classification outcome (Muñoz-Romero, Gorostiaga, Soguero-Ruiz, Mora-Jiménez, & Rojo-Álvarez, 2020; Yao, Cao, Leung, & Liang, 2021). Other post-hoc methodologies deal with the interpretation problem by developing simple proxy models that behave similarly to the original model and implement the perturbation concept at the feature level (Lundberg & Lee, 2017a; Ribeiro, Singh, & Guestrin, 2016). Because the proxy model only approximates the computations of the black box, this approach limits the credibility of the explanations (Mittelstadt, Russell, & Wachter, 2019). Another set of techniques for reducing operation complexity and achieving interpretability makes use of the gradient that is backpropagated from the output prediction to the input layer. These methods create saliency maps by visualizing gradients to highlight areas that the network considers important (Yu, Xiang, Fang, Chen, & Zhu, 2022); nevertheless, solely relying on their explanations, can be misleading (Adebayo et al., 2018). In general, these methods aim to interpret a deep learning model's inference after development and training, which can result in unreliable interpretations (Rudin, 2019).

A different approach to interpretability is the development of ML models that are interpretable by design, e.g., decision trees, lists, and sets (Lakkaraju, Bach, & Leskovec, 2016). Such models are also known as inherently interpretable, and they typically introduce a trade-off between interpretability and accuracy. Since its structure is simpler, its predictive performance may be inferior to that of a more complex black-box model. However, due to the importance of understanding, validating, and trusting ML models in high-risk decision-making domains, this trade-off may be preferable (G. Yang, Ye, & Xia, 2022). CNNs with embedded feature guiding and self-attention mechanisms in their architecture, can also be regarded as inherently interpretable (Sharma & Mishra, 2022). These mechanisms derive interpretations by visualizing saliency maps and CNN features indicating certain concepts on the input image (R. Chen, Chen, Ren, Huang, & Zhang, 2019). Such models, however, typically do not associate the saliency maps with human-perceivable features and do not account for the contribution of these salient regions to the result. Other methods attempt to disentangle features by quantifying the alignment of predefined concepts with learned filters in different layers of a network (Liang et al., 2020); nevertheless, they do not address the direct contribution of the concept representations to the final outcome (Bau, Zhou, Khosla, Oliva, & Torralba, 2017). In addition, the training process of these models demands a considerable amount of manual effort for additional annotations regarding the human-understandable concepts depicted in each image (Barbiero et al., 2022). Approaches extending regular CNNs to encode object parts in deeper convolutional layers, have also been proposed but, they usually lead to the degradation of performance (Q. Zhang, Yang, Ma, & Wu, 2019). Another approach is to leverage the intelligibility and expressiveness of Generalized Additive Models (GAMs) (Hastie & Tibshirani, 1990), which are recognized for their interpretability (Arrieta et al.,

2020). The interpretation provided by a GAM is based on observations that link the effect of each input feature to the predicted output. GAMs are used in a variety of applications to leverage their expressiveness in domains such as healthcare (Y. Cai, Zheng, Zhang, Jiang, & Huang, 2020). GAMs based on Multilayer Perceptrons (MLPs) (Z. Yang, Zhang, & Sudjianto, 2021), were recently proposed for interpretable data classification and regression but these particular models are not tailored for contemporary, CNN-based, computer vision tasks.

More recent interpretable CNN models often utilize information that derives from saliency maps, indicating regions within an image that the model concentrates its attention. These models lacking the ability of explaining how these regions contribute to the predictions. A relevant methodology incorporates interpretable components into a CNN model to explain its predictions (Jung & Kwon, 2021); nevertheless, the provided interpretations are intertwined with predefined edge kernels, and the selection of the color components does not consider any aspects of human perception. In general, there is a paucity of methodologies that have the capacity of explaining the categorization of images based on perceptual qualities, *i.e.*, components such as color and texture represented in a way that people can readily perceive and comprehend (Greisdorf & O'Connor, 2002).

Chapter 3
Saliency Prediction

This section investigates the ability of DL models to estimate visual saliency that approximates that of humans. Eye or gaze tracking has been studied to reveal how visual search and recognition tasks are performed by humans. In the context of medicine, the study of gaze tracking data can provide information that can improve human performance (Lévêque, Bosmans, Cockmartin, & Liu, 2018). The interpretation of medical images is usually challenging and requires domain expertise (Lévêque, Zhang, Parker, & Liu, 2017). The spatial locations within these images, which attract the attention of the medical experts, are characterized as visually salient; however, saliency can be subjective and dependent on the physicians' level of expertise. Gaze data for visual saliency identification are usually collected using specialized eye tracking devices (Judd, Ehinger, Durand, & Torralba, 2009), conventional cameras, such as webcams (Krafka et al., 2016), whereas mouse tracking has also been reported as an interesting alternative for this purpose (M. Jiang, Huang, Duan, & Zhao, 2015).

The problem of gaze estimation based on human eye fixations has been investigated to predict the locations considered as salient by humans, mainly using machine learning-based methods. For example, in (Kummerer, Wallis, Gatys, & Bethge, 2017) a probabilistic model for saliency detection in general images was proposed. That model uses transfer learning from the VGG-19 deep neural network for the prediction of fixation densities. In (Jia & Bruce, 2018), it has been shown that deeper CNN architectures can deliver better results in visual saliency estimation, and proposed a scalable model. In (J. Pan et al., 2017), Pan et al. combined the adversarial loss function with the binary cross entropy to achieve better results in the visual saliency estimation task based on human eye fixations. Recurrent architectures have also been used for that purpose (N. Liu & Han, 2018)(Cornia, Baraldi, Serra, & Cucchiara, 2018). Promising results in the visual saliency prediction, on natural images, have been resulted from methods not only based on machine learning approaches. In particular, a bottom up Graph Based Visual (GBVS) saliency model has been proposed in (Harel, Koch, & Perona, 2007) by Harel et al. The GBVS method consists of two

steps, first forming activation maps on certain feature channels and then normalizing them so that so conspicuities to be highlighted and to admit combination with other maps.

In the context of medical imaging, domain expertise is an important component (Fox & Faulkner-Jones, 2017). Eye tracking studies have been performed for different purposes. Eye trackers have been used to identify how physicians with different levels of experience parse a medical image. Bernal et al (Bernal et al., 2014) used eye-tracking methodologies to investigate the different search patterns between experts and novice physicians, in the context of polyp detection in video colonoscopy.

Jampani *et al.* used eye-tracking to investigate how different imaging methods affect the examination procedure (Jampani, Sivaswamy, Vaidya, & others, 2012). Mendi and Milanova used an algorithmic model that mimics the human viewing behavior by producing visual saliency maps (Mendi & Milanova, 2009). The saliency maps were used for the initialization of an active contour method, for the purpose of medical image segmentation. In another work a computational visual attention model was used for the purpose of hysteroscopy video summarization (Muhammad, Ahmad, Sajjad, & Baik, 2016). However, there has not been an accurate model that is able to predict the gaze behavior of an experienced physician. A thorough literature review investigating eye-tracking applications in the medical domain has been conducted in (Lévêque et al., 2018). It includes studies where eye-tracking was used for identification of expertise, development of training and modeling of visual search patterns, and concludes in that existing computational models cannot sufficiently predict the human gaze behavior.

In section 3.1 a novel methodology for gaze prediction is presented, based on a physicians' eye fixations on images from Wireless Capsule Endoscopy (WCE). The proposed method investigates two deep Convolutional Neural Network (CNN) autoencoder architectures; a basic model, named MedGaze, and an its enhanced version that utilize an additional convolutional block for refining the output along with a post-processing step for more accurate saliency estimation. Both networks are trained to predict the spatial location of the physicians' gaze, on WCE images, in the form of saliency maps. They are inspired from (J. Pan et al., 2017), but with an additional regularization methods being introduced in the decoder part of the network architecture to achieve a more robust performance. Moreover, a novel dataset is presented that was created for the needs of the experiments. The dataset consists of WCE images along with the saliency maps of the medical experts' eye fixations. To the best of our knowledge this is the first application of machine learning algorithm in the context of predicting the spatial location of physicians' gaze in WCE, based on experts' eye fixations.

An extension of this approach is presented in section 3.2 where a novel co-operative training scheme is presented for the improvement of saliency prediction in biomedical images. The proposed methodology incorporates two deep CNNs during the training process, a CNN for predicting the spatial location of the physician's gaze, and a CNN classifier for the purpose of classifying the estimated saliency maps aggregated with their corresponding RGB images, as *normal* or *abnormal*. These two networks are trained in a co-operative manner leveraging a novel

loss function that considers both the accuracy of the saliency maps estimated by the generator model and the error of the CNN classifier, aiming to optimize the saliency map generation performance.

Sections 3.1 and 3.2 are dedicated to approaches that tackle the problem of visual saliency estimation in terms of salient regions. However, the characterization of objects as salient instead of abstract regions is also important and applicable contexts and applications. For instance, salient object detection (SOD) is commonly used for obstacle avoidance in autonomous robot and vehicle navigation (Craye, Filliat, & Goudou, 2016; L. Huang, Li, Li, & Lin, 2019; Klein, Illing, Gaspers, Schulz, & Cremers, 2017). Also, in the context of biomedicine, SOD can be very useful for abnormality detection in medical images (George Dimas, Iakovidis, et al., 2019; Dimitris K Iakovidis, Georgakopoulos, Vasilakakis, Koulaouzidis, & Plagianakos, 2018).

Today, the majority of methods for SOD rely on machine learning approaches, based mainly on Convolutional Neural Networks (CNN). These methods usually determine saliency from plain RGB images (N. Liu & Han, 2016; D. Zhang, Han, Han, & Shao, 2015). However, a limitation of these approaches is that their performance can be easily affected by background objects, which may have similar appearance. This has been tackled by utilizing RGB-Depth (RGB-D) paired information (Qu et al., 2017; N. Wang & Gong, 2019; Zhao et al., 2019). Depth information can effectively describe geometric cues, projected in an RGB image. Hence, it contributes in further improvement of performance and robustness of SOD methods, with regard to applications of object recognition (M. Gao, Jiang, Zou, John, & Liu, 2019; Shah, 2019), and obstacle detection (George Dimas, Diamantis, et al., 2020).

Depth information is acquired using additional sensors, such as stereo camera systems (*e.g.*, Intel RealSense), structured light based (*e.g.*, Microsoft Kinect), Light Detection and Ranging (LIDAR) systems *etc*. Such sensors can provide sufficient information about depth; however, their use introduces an additional overhead to the overall cost and flexibility of a system.

Recently machine learning models have been proposed for monocular depth estimation, with a performance approximating that of RGB-D sensors (Alhashim & Wonka, 2018; Chaney, Bucher, Chatzipantazis, Shi, & Daniilidis, n.d.; Tateno, Tombari, Laina, & Navab, 2017). However, these methods are yet to be assessed with respect to their capacity of representing depth effectively for SOD.

Several CNN-based methods have been proposed to deal with SOD in images obtained by RGB-D sensors. Han *et al.* (Han, Chen, Liu, Yan, & Li, 2017) proposed a deep CNN model composed of three separable CNNs, in order to generate pixel-level saliency maps. More specifically, an RGB image and its corresponding depth image are fed into two detached networks to extract the RGB and depth view respectively. Transfer learning techniques such as deep supervision in hidden layers and task-relevant initialization, are integrated into the network which processes the depth cue for enhancing the generated depth representation. Then, a third CNN receives as input the above acquired views, to automatically combine both the RGB and the depth representation in an

optimal way. Furthermore, to efficiently capture the high–level semantics, a global structural loss is adopted during training process.

Wang and Gong (N. Wang & Gong, 2019) suggested a dual-stream CNN to produce saliency maps. This approach enables a synchronous processing of RGB and depth images. Each branch computes a saliency map of either the RGB or the depth image and simultaneously extracts high level features from the input images. A trainable saliency fusion module is used to efficiently incorporate the generated saliency maps by taking into consideration the extracted high-level features. Chen *et al.* (Hao Chen, Li, & Su, 2019) also adopted a double stream architecture to generate saliency maps. Initially, each stream which contains two branches, for global and local feature extraction, is trained independently. Afterwards, both RGB and depth local features are directly combined, whilst global branches are merged in a fully connected layer. Then, the whole network is trained again in an end-to-end manner and the predictions corresponding to the local and global branches are summed up to construct the final saliency map.

In another work, Chen and Li (Hao Chen & Li, 2018) introduced complementarity-aware fusion (CA-Fuse) modules to take full advantage of the complementary information included in the pairs of color and depth images. The suggested approach interposes CA-Fuse modules between the parallel convolutional layers of a two-stream architecture. These trainable modules select color and depth features and efficiently combines them, and they progressively fuse their predictions to facilitate the construction of saliency maps. Zhao *et al.* (Zhao et al., 2019) developed a so-called fluid pyramid integration model enabling a CNN architecture to exploit both cross-modal information and multi-scale features for saliency map generation. A contribution of that study was the automatic enhancement of depth information before its incorporation in the fluid pyramid integration model, by leveraging a contrast prior.

A three-stream attention-aware network was proposed by Chen and Li (Hao Chen & Li, 2019) for saliency map generation. Apart from the two streams usually used for extracting color and depth features, in that study, an additional stream is used in parallel. This stream is used to discriminate representative RGB and depth views, by receiving as input, corresponding concatenated features extracted from each level of the depth and RGB streams. Then, it utilizes these views to learn supplementary RGB-D features. Eventually, the three streams are combined in a shallow CNN to produce the final saliency map.

Fan *et al.* (Fan, Lin, Zhang, Zhu, & Cheng, 2020) have also proposed a three-stream architecture, called D^3Net, which unlike the previous one, is composed of three, separately trained, convolutional sub-networks. Each of these sub-networks aims to estimate accurate SOD by receiving as inputs RGB, depth and cross-modal RGB-D tensors, respectively. Hence, Depth, RGB and cross-modal saliency maps are produced from each stream. Moreover, a histogram-based depth depurator unit is introduced during the inference process to discard the low-quality depth maps and fuse the remaining maps to predict the final saliency map.

The previous methodologies depend on the complementary depth information, derived by additional sensors, which introduces an implementation and cost overhead for real-world applications. Additionally, the most recent RGB-D SOD architectures are based on three networks to achieve high SOD accuracy. To address this challenge, the study that is presented in section 53.3: i) proposes a novel monocular salient object detection model, abbreviated as MonoSOD, where the depth information is estimated by a CNN model; ii) presents an improved architecture of a SOD model, characterized by lower computational complexity than that of the respective state-of-the-art model; iii) introduces an additional, trainable, post-processing step, where the output of the SOD process is refined for improved accuracy; iv) compares several pretrained CNN-based depth estimation models with sensor-based ones in the context of SOD. The proposed approach is based on a model with two branches, capable of both predicting depth and estimating the saliency of an object. This approach simplifies the hardware and software requirements for SOD.

3.1 Visual Saliency Estimation on Medical Images

Basic Saliency Model

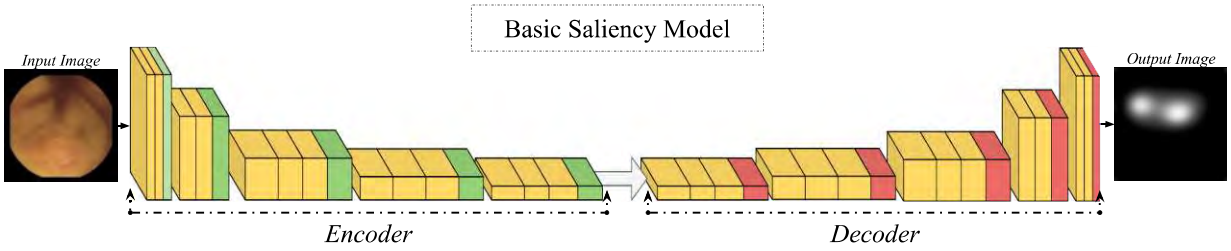


Figure 3.1. Overview of the basic saliency model architecture

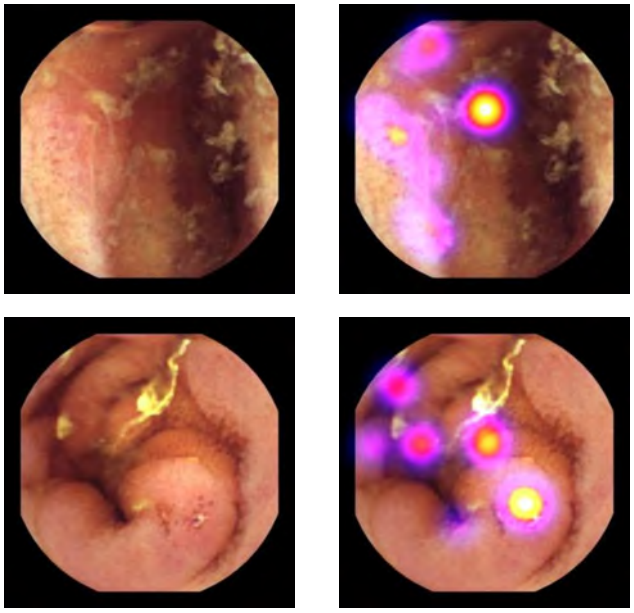


Figure 3.2. Illustration of the gaze saliency maps based on the physician's eye fixation. (a) The RGB WCE image used as a visual stimulus. (b) A physician's eye fixations overlaid on the visual stimulus. The yellower the color the more salient the location of the image.

The proposed methodology for gaze prediction is based on a deep Convolutional Neural Network (CNN) trained to learn and estimate saliency maps that are based on physicians' eye fixations on WCE images (Figure 3.1). The network receives as input an RGB WCE image and outputs a saliency map that approximates the medical experts' gaze on the respective image. The details about the CNN architecture and its training, as well as the methodology for the dataset creation are described in the following subsections.

Our model consists of two parts, an encoder, which is used to compress the input volume to a latent-space representation and a decoder, which is employed to reconstruct the target objective using that latent-space representation. The architecture of the proposed model is illustrated in Figure 3.1.

The encoder part of the model has a similar architecture with the VGG-16 CNN model (Simonyan & Zisserman, 2014). The first two blocks of the model consist of two convolutional (orange and light orange) layers and a max pooling layer (pink). The rest of the blocks consist of three convolutional layers (light yellow) and a max pooling layer. This model is used as an encoder, with the final max pooling and fully connected (FC) layers of the original VGG-16, removed. The max pooling layers of the encoder are used to down-scale the feature maps. The weights used are those of a VGG-16 network trained on ImageNet dataset (J. Deng et al., 2009).

The decoder part of the model has the same architecture with the encoder but with a reverse order of blocks. Also, the max pooling layers have been replaced with up-sampling layers. The up-sampling layers are used to increase the size of the output to match the dimensions of the input image. The first two blocks consist of three convolutional layers (light green) and an up-sampling layer (dark orange). The following two blocks consist of two convolutional layers (green and dark green) and an up-sampling layer. An additional, final, point-wise convolutional layer (red), *i.e.*, a convolutional layer with a kernel of size 1×1, has been added, with a sigmoidal activation function for the saliency map generation. The point-wise convolution has been chosen for a more precise saliency estimation, so forth, each pixel is being examined if it is salient instead of a neighborhood. The values of the saliency map can be expressed as probabilities of a pixel being salient, so the sigmoidal activation function which has a response range in the interval of [0, 1] is the best fit for that purpose. In addition, the weights of the decoder are randomly initialized. The activation functions of the convolutional layers of the encoder-decoder network are the Rectified Linear Unit (ReLU) and all, except from the last layer, use kernels of size of 3×3. The last convolutional layer of the decoder has a kernel with a size of 1×1 and as an activation function the sigmoidal.

The network receives as input an RGB WCE image with dimensions of 96 × 96 × 3, *i.e.*, with a height and width of 96 pixels and 3 color channels normalized to the interval of [0, 1]. The objective of the model is to estimate a saliency map based on physicians' eye fixations. During the training process the saliency maps are also normalized to the interval of [0, 1] and their dimensions are of 96 × 96 × 1. Since the values of the saliency maps vary in the interval of [0, 1] they can be interpreted as probabilities of pixels being observed by the physicians. Furthermore, because of the peripheral vision, more pixels may be observed around of the fixation points, and that makes it appropriate to apply a point-wise convolutional operation on the final layer. In that context, a loss function that fits the solution of the aforementioned task, is the binary cross entropy (BCE) loss function. BCE is calculated as the average of the individual BCEs of all pixels in an estimated saliency map. A saliency map S can be formally expressed as follows:

$$S(x, y): N \times M \rightarrow [0, 1] \tag{3.1}$$

Then we can calculate the BCE between the ground truth S and the estimated S' saliency maps as follows:

$$L_{BCE} = -\frac{1}{N \cdot M} \sum_{i=0}^{N} \sum_{j=0}^{M} S_{ij} \log\left(S_{ij}^{'}\right) + (1 - S_{ij}) \log\left(1 - S_{ij}^{'}\right) \tag{3.2}$$

The training algorithm that was used was AdaGrad because it greatly improves the robustness of SGD and is effective for the training of large-scale neural networks. The learning rate value was 3×10^{-4} and L_2 weight regularization was applied on all the convolutional layers with $\lambda = 1 \times 10^{-4}$. To achieve a more robust model, we introduced to the decoder part of the architecture the dropout regularization algorithm. The dropout algorithm leads to units that are robust and are dependent free from the activation of other independent units (Baldi & Sadowski, 2014). Thus, with the

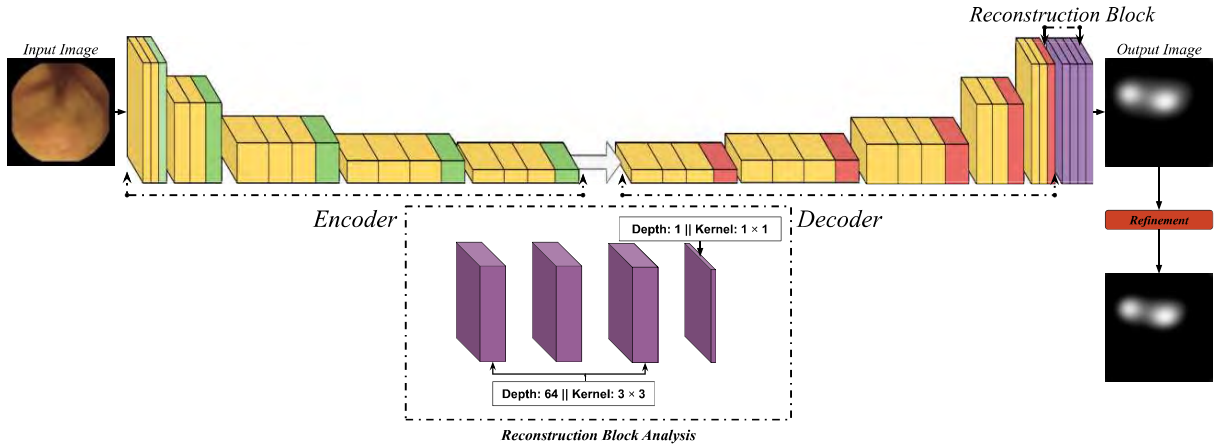
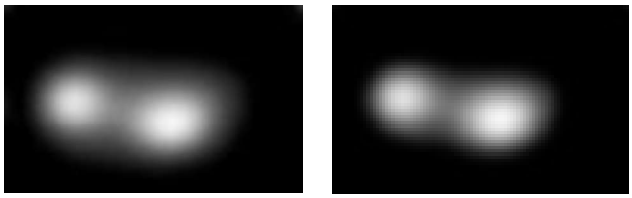


Figure 3.3. Overview of the enhanced saliency model architecture.



(a) (b)

Figure 3.4. Illustration of the refinement effect of the estimated saliency map. (a) Estimated saliency map, (b) refined saliency map of (a).

application of the dropout algorithm on our network, we achieve a more generalized approach to the problem we intend to tackle.

Enhanced Saliency Model

The enhanced architecture that was developed has the same architecture with the basic model with an exception to the decoder model. In detail, after the final convolutional block of the decoder, a reconstruction convolutional block has been added aiming towards a more thorough analysis of the latent representation of the encoder, and a more precise restoration of the ground truth saliency map. The size of the final reconstruction block can affect the quality of the results produced by the model. To maintain low complexity and high reconstruction quality, 3 convolutional layers followed by a pixel-wise convolution for the final saliency prediction was experimentally determined to achieve satisfactory results. Each one the three convolutional layers of the reconstruction block is composed of the same number of kernels, equal to the respective number utilized in the convolutional layers of the block before the reconstruction module. The point-wise convolution layer uses a kernel with a size of 1×1 and thus examines the saliency on a pixel level instead of a broader neighborhood. The activation function of that final layer has been chosen to be the sigmoidal since saliency can be interpret as the probability of a pixel being salient, *i.e.*, within the [0, 1] interval. The activation function of the convolutional layers, of both the encoder and the decoder, is the Rectified Linear Unit (ReLU) and the size of the convolutional kernels is 3×3. Exception is the final convolutional layer which is assigned with the sigmoidal activation function and has a kernel size of 1×1.

After the saliency map estimation, we deploy an additional step for further improvement of the saliency map by removing salient values with low probability of occurrence. For this step, a parametric Heaviside function, H is utilized:

$$H(S(i,j)) = \begin{cases} 1, & \text{if } \sum\sum S(i,j) > a \\ 0, & \text{if } \sum\sum S(i,j) \leq a \end{cases} \quad (3.3)$$

where a, $S(i, j) \in [0, 1]$, a is a trainable parameter and $S(i, j)$ a value of a pixel of the produced saliency map, S. The function H considers a neighborhood around a target pixel at (i, j). The application of H on S discards salient pixels with low probability of occurrence, refining the saliency map for its further exploitation on other applications. An illustrative example of this model is depicted in Figure 3.3. An example of the application of Eq. (3.3) on a saliency map can be seen in Figure 3.4.

Eye Fixation Dataset

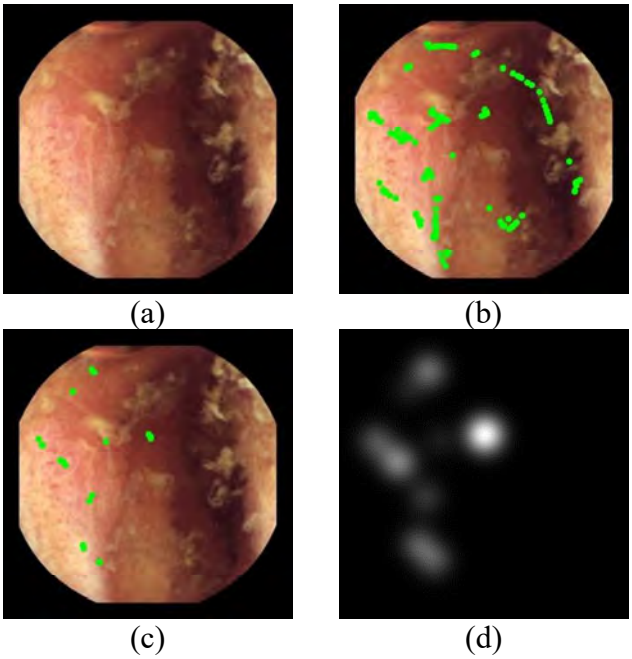


Figure 3.5. A qualitative comparison among the ground truth and estimated gaze density maps. (a) The RGB WCE image used as input to the network; (b) the ground truth saliency maps based on physicians' eye fixations; (c) the estimated saliency maps.

To train a CNN, it is of major importance to find a dataset with a sufficient amount of data. In the context of eye fixations, on WCE images, to our best of our knowledge, there is not any openly available dataset for WCE, so we created our own dataset, using the publicly available WCE image dataset KID (Dimitris K Iakovidis & Koulaouzidis, 2015)(A. Koulaouzidis et al., 2017). The KID dataset is based on WCE anonymized images and videos, with high quality annotations made by experts in the field.

A physician with expertise in the field of WCE, examined the images from the KID dataset. Eye tracking was performed using The Eye Tribe© eye tracker. The physician was instructed to examine each image presented, and once he was confident for a diagnostic assessment, to move forward onto the next one. The data that was captured w, the tracking points of the eyes in terms of pixel coordinates and the timestamp of each point. During the eye-tracking data acquisition process, the physician was not allowed to move from his initial position in which the eye tracker has been calibrated for the maximum accuracy. However, since it is practically not possible for the physician to be absolutely still, in order to maintain the accuracy of the experiment sufficiently high, the eye-tracker was re-calibrated every 30 images to avoid inconsistencies.

The data points that were recorded during the eye tracking process were categorized to saccade and fixation points. We focus on fixation points because during saccades, little to none visual processing can be achieved (Salvucci & Goldberg, 2000). We used the velocity based, Velocity Threshold Identification (I-VT) algorithm for the fixation estimation because of its accuracy and its implementation ease (Salvucci & Goldberg, 2000). An illustration of the eye tracking data points before and after the fixation estimation can be seen in Figure 3.5.

For the target saliency map creation, we applied the two-dimensional Gaussian Function (Bernal et al., 2014) on each fixation point that was estimated by the I-VT algorithm. The parameter σ of the Gaussian Function was set to be the time difference, Δt, between the current fixation point and the next one, formally expressed as:

$$G(x, y) = \frac{1}{2\pi\Delta t^2} e^{-\frac{x^2+y^2}{2\Delta t^2}} \tag{3.4}$$

In that manner, we are considering the peripheral vision of the physician.

Experiments and Results

To evaluate the proposed methodology we considered the Judd implementation of the Area Under Curve (AUC-J) (Riche, Duvinage, Mancas, Gosselin, & Dutoit, 2013). AUC-J is calculated by treating the saliency maps as binary classifiers by separating the negative and positive samples in different thresholds. Given a threshold, the positive rate is described by the pixel-values of the saliency map that are above it, at the fixation locations. Accordingly, the false positive rate is determined by pixel-values that are above a given threshold, at non-fixation locations described by the pixel-values of the saliency map that are above it, at the fixation locations. Thus, the false positive rate is determined by pixel-values that are above a given threshold, at non-fixation locations.

For the evaluation of our model we exploited 5 out of the 7 categories of the KID dataset. Namely, the inflammatory, vascular, polypoid, normal oesophagus and colon image categories were used concluding to 1025 images in total. In detail, in our dataset there were 227 images depicting inflammatory lesions, 303 images of vascular lesions, 44 images of polypoid lesions and 451 images of normal tissue from the colon and oesophagus. We used the 5-fold stratified Cross-Validation (CV) procedure for the training instance, where the images of each category were split into training, validation and testing set with proportions of 60%, 25% and 15% respectively. The input shape of the data was empirically determined to be $96 \times 96 \times 3$. The training, validation and testing data were augmented to increase the number of samples in the dataset for the training and evaluation. The images were augmented with respect to their orientation and the augmentation was performed by rotating the images within the interval of [0º, 80º] with an angle step of 20º. Additionally, the early stopping method was applied monitoring the validation performance during the training process. The proposed model was trained using the Adagrad algorithm with an initial learning rate value of 3×10^{-4}.

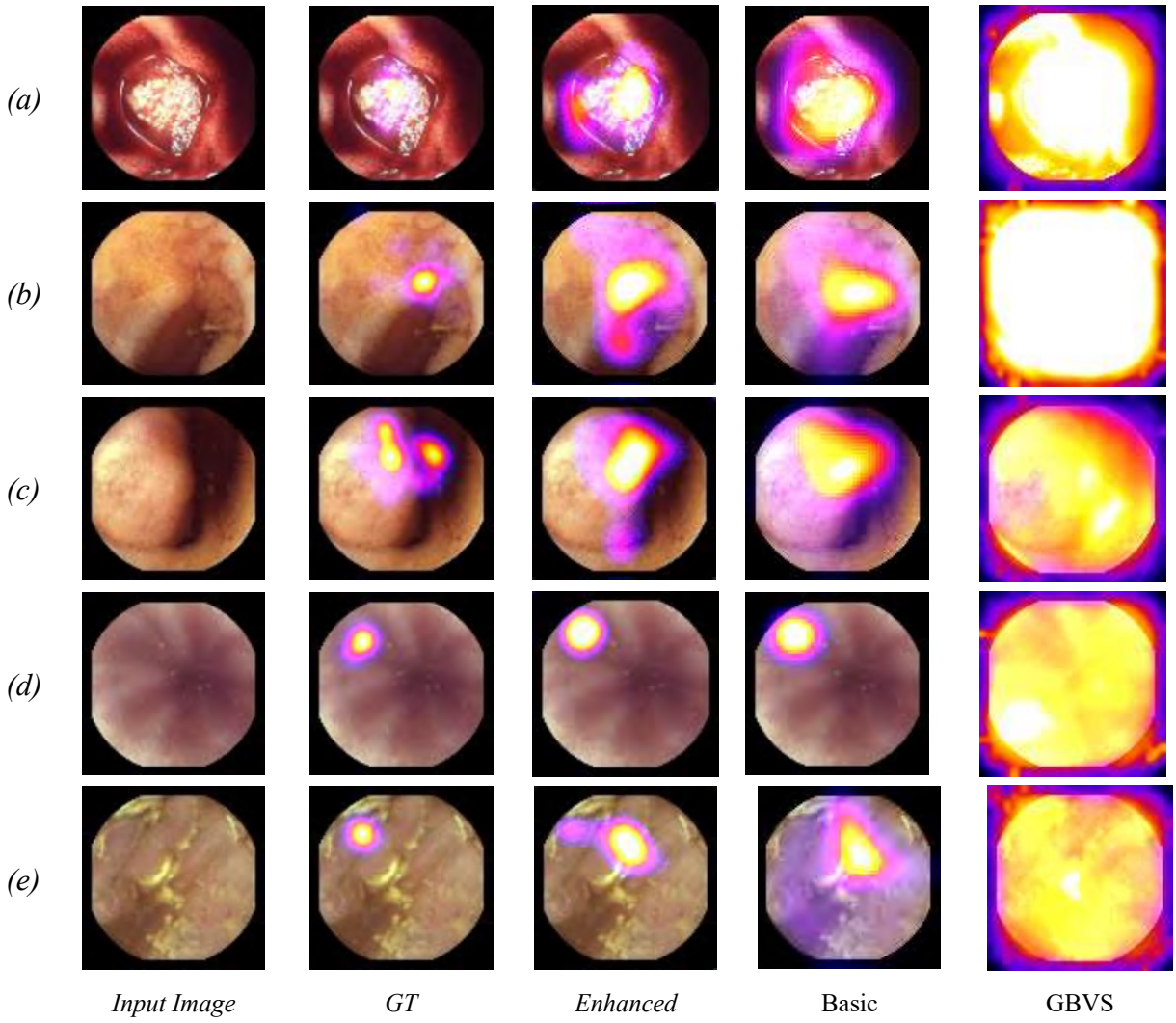


Figure 3.6. Qualitative comparison among the proposed and other saliency estimation models with applications on the biomedical domain. Each row represents a different category : (a) the vascular lession, (b) the inflammatory lession, (c) the polypoid lession, (d) the normal colon and (e) the normal oesophagus category

The experimental results derived from the evaluation of the proposed methodology are summarized in Table 3.1. According to Table I the average AUC-J achieved by the introduced CNN model was 0.718. In detail, the AUC-J scores for the normal image categories of the colon and the oesophagus were 0.652 and 0.863 respectively. As for the abnormal image categories of vascular, inflammatory, and polypoid lesions the average AUC-J scores for all the iterations of 5-fold CV were 0.662, 0.695, 0.719 accordingly.

Moreover, a comparison was conducted between the proposed method and relevant state-of-the-art methods for saliency prediction on biomedical images. Particularly, the GBVS graph model (Harel et al., 2007) and MedGaze CNN model (G. Dimas, Iakovidis, & Koulaouzidis, 2019). The

GBVS model achieved average AUC-J score of 0.591 in all image categories. The MedGaze model resulted in an average AUC-J score of 0.693. In general, the proposed model outperformed both

Table 3.1. AUC-J scores on each WCE image category

Data	Models		
	Enhanced Model	*Basic Model*	*GBVS[23]*
Inflammatory	0.715	0.656	0.589
Vascular	0.683	0.665	0.617
Polypoid	0.720	0.694	0.617
Normal Oes.	0.877	0.826	0.574
Normal Col.	0.636	0.626	0.562
Average	0.726	0.693	0.591

the GBVS and MedGaze model for all the image categories. All models compared present standard deviation of the order of ~10^{-2}. Additionally, a qualitative performance assessment is presented in Figure 3.6. As it can be observed the proposed model managed to predict more accurately the eye-fixation regions of the physicians when compared to the other two models.

3.2 Co-Operative CNNs for Saliency Detection on Biomedical Images

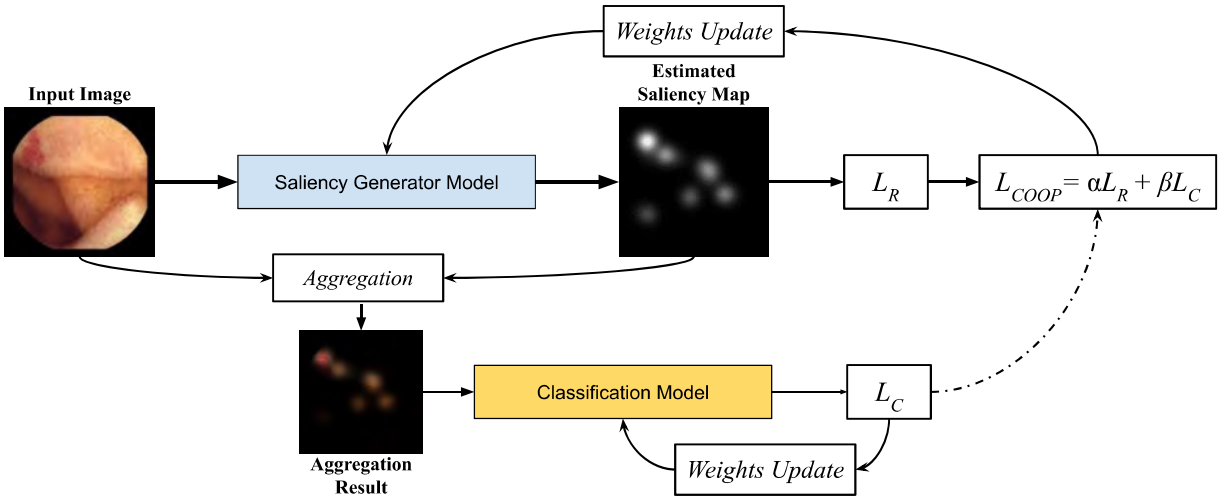


Figure 3.7. Overview of the proposed methodology

The gaze prediction methodology proposed in this section, is based on the synergy of two CNN models trained in a co-operative way. The first CNN model has an autoencoder architecture and it is tasked to estimate saliency maps approximating the physician's eye fixations on WCE images. The second model is a CNN classifier, which receives as input the aggregation of an estimated saliency map with its corresponding RGB image. This model is tasked to classify the aggregation

result as normal or abnormal. Both models are based on the VGG-16 CNN model pretrained on the ImageNet dataset (Simonyan & Zisserman, 2014), (J. Deng et al., 2009). These CNNs are trained in parallel. The classification error is back-propagated only to the classifier model whereas both the classification and saliency estimation error are fused together using a joint loss function that is utilized to update the weights of the saliency map generation model. An overview of the proposed methodology is illustrated in Figure 3.7.

The model that was employed for tackling the task of saliency prediction was the one described in section 3.1. The classifier model used in the proposed methodology is a CNN model connected with an FC part. The CNN model architecture is identical to that of VGG-16 model with an exception to the FC part. The FC part that is used to our classifier comprises 3 layers. Each of the first two layers have 100 neurons and the last layer that is tasked with the label prediction has 1 neuron, since our classification problem is defined by two labels, *i.e.*, normal and abnormal (0 and 1 respectively). The activation function that was used for all neurons was the sigmoidal activation function.

Similar to the Generative Adversarial Network (GAN), the proposed training scheme incorporates two models (Goodfellow et al., 2014). In the case of GANs, the generator and discriminator networks are engaged in a competitive min-max game, where a generative network, G, is trying to maximize the error of the discrimination network D and vice-versa. On the other hand, in the proposed training scheme the saliency generator model tries to produce saliency maps that minimize the classification error of the classifier model. At the same time, the classifier provides feedback to the generator model regarding the classification error to assist the change of the generator weights for more accurate saliency map prediction.

This process has been inspired by the examination process performed by physicians' eyes. During this process, both visual and cognitive processes are performed, and physicians' decisions are based on the information extracted from the regions corresponding to their eye fixations (Just & Carpenter, 1984). Since these regions are mainly considered for the medical decision making, instead of using the whole images, only the salient regions of the images are sufficient to train a classifier for the task of categorizing normal and abnormal findings. These regions can be isolated by aggregating the saliency map S and its corresponding image I_{RGB} by multiplication (Figure 3.7).

Each epoch of the proposed training process can be summarized into three steps. The first step of the training is the saliency map estimation and the assessment of the reconstruction loss, L_R. As a reconstruction loss, the binary cross-entropy (BCE) loss function is used. At the second step, the estimated saliency map S is multiplied with the input image I_{RGB} providing an image I_S where only the salient regions of image I_{RGB} are visible. Subsequently, I_S is propagated to the classifier network C, which is tasked to classify the input as normal or abnormal. Once again, the classification loss, L_C, is assessed using the BCE. Once both L_R and L_C are calculated, the latter, *i.e.*, L_C, is backpropagated to the classification model C, whereas a joint loss function that we refer to as co-operative loss (L_{COOP}) is backpropagated to the saliency generator model G_S. L_{COOP}, incorporates both L_R and L_C as follows:

$$L_{COOP}(L_R, L_C) = aL_R + \beta L_C \quad (3.5)$$

where α and β coefficients used as weights determining the significance of each loss factor. Then, in the third step L_{COOP} is backpropagated to the saliency generator model G_S. With the use of the joint loss the network is forced to alter its weights towards the generation of accurate saliency maps that can be used to increase the performance of image classification. As it is presented in the following section, this training leads to the generation of more accurate saliency maps.

Experiments and Results

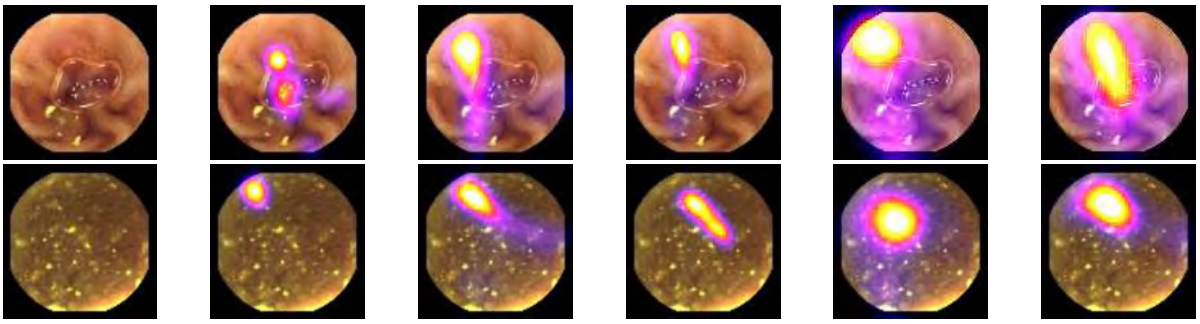


Figure 3.8. Qualitative comparson of saliency maps generated by various methods.

Table 3.2. AUC-J scores on each WCE image category

Data	Models			
	Proposed	*GAN*	*Basic*	*Enhanced*
Inflam.	0.76	0.66	0.66	0.72
Vascular	0.72	0.67	0.67	0.68
Polypoid	0.77	0.69	0.69	0.72
Normal Oes.	0.87	0.86	0.82	0.87
Normal Col.	0.69	0.72	0.626	0.64
Average	0.76	0.72	0.69	0.73

For the training the dataset described in 3.1. The input RGB WCE images were downscaled to 96×96 pixels. The output of the saliency generator network was a single-channel saliency map with dimensions of 96×96 pixels, with pixel values in the interval of [0, 1]. The classification network receives the saliency generator input image multiplied with the generated saliency maps to isolate the salient regions, so the input image dimensions of the classification network were also 96×96. Both the input images of the generator and the classification network were also normalized in the interval of [0, 1]. The training algorithm that was applied on both the saliency generator network and the classifier was the AdaGrad with a learning rate of 3×10^{-4}. The dropout algorithm was also used, with dropout-rate of 0.5, along with the L_2 regularization ($\lambda = 1\times10\text{-}4$) on all the convolutional layers of the decoder. Furthermore, the training, validation and testing data were augmented to increase the number of images in the dataset for the purpose of training and evaluation. The images were augmented with respect to their orientation and the augmentation was performed by rotating the images within the interval of [0º, 80º] with an angle step of 20º.

To assess the performance of saliency generation methods the Judd implementation of the Area Under Curve (AUC-J) was considered, as a widely accepted measure (Riche et al., 2013). AUC-J is calculated by treating the saliency maps as binary classifiers by separating the negative and positive samples in different thresholds. Given a threshold, the positive rate is described by the pixel-values of the saliency map that are above it, at the fixation locations. Accordingly, the false positive rate is determined by pixel-values that are above a given threshold, at non-fixation locations.

The average score among all data categories of our proposed cooperative model in terms of AUC-J was 0.76 (Table 3.2). Additionally, the evaluation included state-of-the-art models for saliency predictions on medical images, *i.e.*, the basic and enhanced models proposed in section 3.1. Since the proposed co-operative training procedure resembles the GAN training scheme, for comparison, the proposed model was trained with the adversarial loss, according to (J. Pan et al., 2017). For the training of the GAN, the generator architecture and input were kept the same, whereas the discriminator input changed to the concatenation of the input image I_{RGB} with the corresponding saliency map.

In terms of average performance, in most cases the proposed cooperative model provided a higher score in terms of AUC-J. A detailed summary describing the performance of each network that was used in this evaluation process can be seen in Table 3.2. Furthermore, a qualitative comparison of the saliency maps generated by the methodologies incorporated in the evaluation process is presented in Figure 3.8.

3.3 MonoSOD: Monocular Salient Object Detection

The architecture of MonoSOD is illustrated in Figure 3.9. It consists of two autoencoder CNN branches. The first one, implements depth prediction, and the second one implements SOD based on the predicted depth. In this study we investigate two backbone architectures for depth prediction, namely *DenseDepth* (Alhashim & Wonka, 2018) and *Monodepth2* (Godard, Mac Aodha, Firman, & Brostow, 2019). The SOD network enhances the RGB-D sub-network proposed by Fan *et al*. (Fan et al., 2020) by incorporating a SOD refinement step. More specifically, the input of this network is a tensor consisting of an RGB image and the respective depth map generated by the depth prediction branch. This tensor, representing the joint RGB-D information and is subsequently processed and propagated through the network. As a result, an initial SOD is estimated, which is then refined by a trainable filtering scheme, enhancing the detection accuracy of the salient object.

A CNN model capable of predicting a depth map D_M from an RGB image I_{RGB}, can be considered as an approximation function $f(I_{RGB}) = D_M$, with $f: I \rightarrow D$. Such function, maps values of RGB

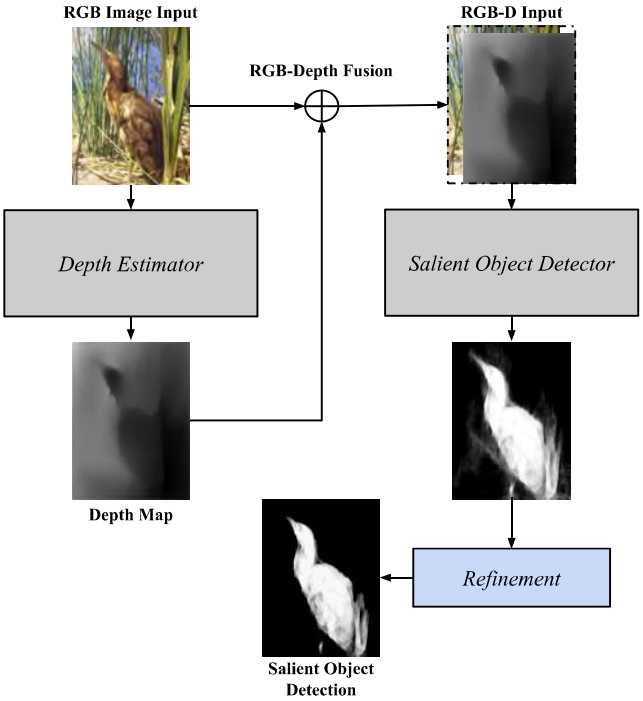


Figure 3.9. MonoSOD two-network architecture.

images to depth values, considering *I* as the input space of all RGB color images, and *D* as the depth space of the images in *I*.

For the purposes of our study, we consider two state-of-the-art CNN-based models as depth predictors, namely *DenseDepth* (Alhashim & Wonka, 2018) and *Monodepth2* (Godard et al., 2019). These networks have a similar U-Net architecture with different backbone networks. Another difference is that *DenseDepth* is a supervised model, whereas *Monodepth2* follows a self-supervised learning paradigm. These models are briefly described below.

DenseDepth: The architecture of *DenseDepth* model consists of an encoder and a decoder network. The encoder network aims to map an input image I_{RGB} to a latent representation and has the same architecture with DenseNet-169 (G. Huang et al., 2017). The weights of the encoder are initialized by training the model, for a classification task, on ImageNet dataset (J. Deng et al., 2009). The latent representation of that I_{RGB} image is then propagated to a decoder network. The decoder consists of up-sampling layers followed by convolutional layers and skip-connections from the encoder. The decoder outputs a corresponding depth map in half the resolution of the input image. The output depth map is then resized with bilinear up-sampling to match the dimensions of the input RGB images.

Monodepth2: The encoder of *Monodepth2* is a ResNet18 (He et al., 2016), similarly to the *DenseDepth*, its weights were initialized by using the ImageNet dataset. The decoder network that Monodepth2 uses, is similar to that of (Chaney et al., n.d.), with sigmoid activation function at the final layer, responsible for the output and exponential linear units (Clevert, Unterthiner, & Hochreiter, 2015) non-linearities elsewhere. In addition, the decoder network employs reflection padding instead of zero padding which reduces the border artifacts in the produced depth map. *Monodepth2* is considered as self-supervised because it is trained to predict the appearance of a target image from the viewpoint of another image.

The encoder of MonoSOD extends the VGG-16 model (Simonyan & Zisserman, 2014) with an additional convolutional block, to preserve, as much as possible, the semantic information of the input images. The VGG-16 network consists of five convolutional blocks and a fully connected

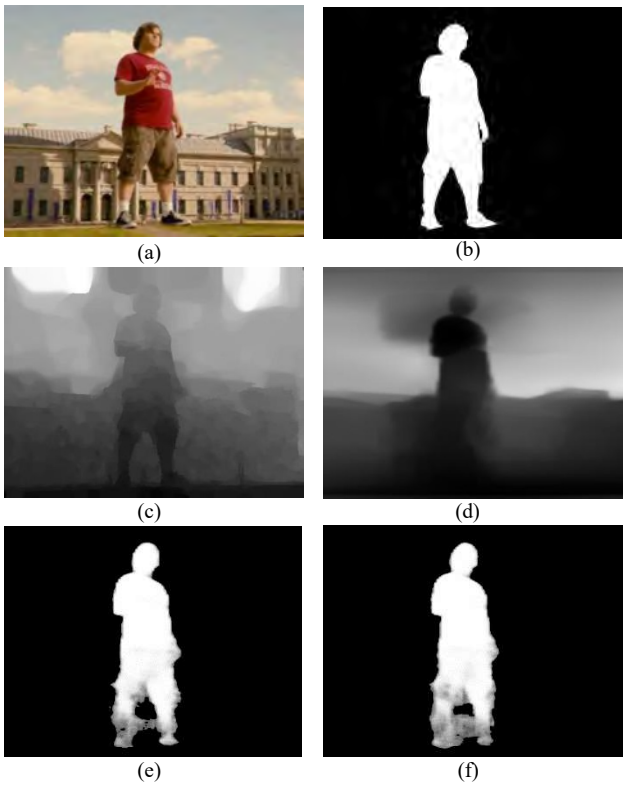


Figure 3.10. MonoSOD vs. RGB-D SOD. (a) RGB input image; (b) Ground truth (GT); (c) RGB-D depth map; (d) Predicted depth; (e) Detected salient object using MonoSOD; (f) Detected salient object using RGB-D SOD.

layer. The first two blocks are composed of two convolutional layers followed by a max pooling layer. The remaining blocks are consisted of three convolutional layers followed by a max pooling layer. Both the final max pooling layer and the fully connected layer of the original VGG-16 network have been removed from the model. An additional convolutional block composed of two convolutional layers and an average pooling layer has been added to expand the architecture of VGG-16 network. The convolutional layers of the 6th block utilize 3×3 kernels and have a depth size of 32. All the convolutional layers of the encoder utilize convolutional kernels of size of 3×3 and the Rectified Linear Unit (ReLU) as activation function.

The decoder of the network has the same architecture with the first five blocks of the encoder but in a reverse order and simplified structure. In detail, all the convolutional blocks of the decoder consist of single convolutional layers with 3×3 and 32 kernel and depth size respectively. In the decoder, all the max-pooling layers have been removed and replaced with up-sampling layers. To attain richer features, the feature maps deriving from the first five layers of the encoder, are concatenated with the respective feature maps of the decoder. Before the concatenation, the feature maps of the decoder undergo a 1×1 convolution operation in order to reduce the number of channels and maintain the overall number of channels equal to four after the concatenation process.

The final block of the decoder can be described as a prediction block. This block consists of two convolutional layers with a kernel of 3×3 dimensions and a depth size of 32 as the final convolutional layers of the encoder, and one convolutional layer with a kernel of size 1×1. In this final convolutional block, the first two convolutional operations utilize the ReLU activation function. Since the values of the expected output are within the range of [0, 1] the sigmoidal activation function is used for the prediction of the saliency map, which depicts the respective salient object.

For the training of the network, the Binary Cross Entropy (BCE) function was adopted. BCE is estimated as the average of the individual BCEs of all pixels in an estimated saliency map *s*, formally expressed as:

$$L_{BCE}(s,g) = -\frac{1}{N \cdot M}\sum_{i=1}^{N}\sum_{j=1}^{M} g_{ij}\log(s_{ij}) + (1-g_{ij})\log(1-s_{ij}) \tag{3.6}$$

where $s_{ij} \in [0,1]$ and $g_{ij} \in [0,1]$ are the values of *s* and the ground truth (GT) *g*, at a pixel coordinate (i, j) of *s* and *g*, respectively, where $i = 1, 2, 3\ldots N, j = 1, 2, 3\ldots M$ and *N, M* are the dimensions along the vertical and horizontal axis of the SOD map respectively.

After the initial SOD, we introduce an additional refinement step. For the refinement, we introduce a parametric Heaviside function, H_p formally expressed as:

$$H_p(x) = \begin{cases} x, & \text{if } x > a \\ 0, & \text{if } x \le a \end{cases} \tag{3.7}$$

where $a, x \in [0, 1]$, *a* is a trainable parameter and *x* a value of a pixel of the produced SOD map, *s*. The function H_p is employed to refine the saliency map in a convolutional manner, considering the local neighborhood around a pixel at a coordinate (i, j). This operation can be expressed as:

$$\tilde{s}(i,j) = H_p\left(\frac{1}{m \cdot n}\sum_{k=-a}^{a}\sum_{l=-b}^{b} s(i-a, j-b)\right) \tag{3.8}$$

where *s* is the SOD map, initially estimated by the model, $\tilde{s}$ is its refined counterpart while *m* and *n* denote the width and height of the local neighborhood. This refinement process can have a significant impact to SOD accuracy as indicated by the results presented in the experiments and results section.

Experiments and Results

The experimental evaluation was performed on 3 benchmark RGB-D datasets for SOD, namely, NLPR (H. Peng, Li, Xiong, Hu, & Ji, 2014), NJU2K (Ju, Ge, Geng, Ren, & Wu, 2014), and STERE (Niu, Geng, Li, & Liu, 2012). Both NLPR and STERE datasets are composed of 1000 images, whereas NJU2K has approximately 2000 images. The depth information in these datasets, was estimated either using stereo cameras (in NJU2K, STERE) or it was acquired using the Microsoft Kinect sensor (in NLPR).

To conduct a fair comparison with other methodologies we have followed the data splitting approach indicated by (Fan et al., 2020), *i.e.*, both NLPR and NJU2K datasets were split in two parts. In total, 700 image-depth pairs from NLPR and 1485 from NJU2K dataset were selected to construct the training and validation data partition for the training of MonoSOD. The remaining 300 image pairs from NLPR and 500 from NJU2K dataset, and the whole STERE dataset were used for testing purposes.

The size of the images used for training was 224×224 pixels. MonoSOD was trained using the Adam algorithm with an initial learning rate value of $1e^{-4}$. The model was trained using the early stopping technique monitoring the loss of the validation partition. The weights of the model was initialized using the method described in (He, Zhang, Ren, & Sun, 2015). The training images were augmented by horizontally flipping (Perez & Wang, 2017), (Fan et al., 2020). DenseDepth and MonoDepth2 were trained using the NYU Depth v2 (Silberman, Hoiem, Kohli, & Fergus, 2012) and KITTI (Geiger, Lenz, & Urtasun, 2012) datasets respectively.

For conducting the evaluation of the proposed model, four evaluation metrics were selected, namely, the Mean Absolute Error (MAE) (Perazzi, Krahenbuhl, Pritch, & Sorkine-Hornung, 2012), the max *F*-measure (F_β) (Borji, Cheng, Jiang, & Li, 2015), the Structure measure (*S*-measure, S_a) (Fan, Cheng, Liu, Li, & Borji, 2017), and the Enhanced alignment measure (*E*-measure, E_ξ) (Fan et al., 2018).

MAE: Enables the quantification of the deviation between estimated and GT maps. The mean absolute error between a real-valued saliency map s and a binary GT g for all image pixels is calculated as follows:

$$MAE = \frac{1}{N} |s - g| \tag{3.9}$$

where N is the total number of pixels. MAE estimates the approximation degree between the saliency and the GT map, and it is normalized in the interval [0,1]. Generally, the error is proportional to the size of the object. Thus, smaller objects are typically correlated with smaller errors and vice versa. Although, this measure provides a direct estimate of the errors, it completely ignores their spatial properties.

F-measure: This metric is regarded as a region-based similarity metric. *F*-measure is a harmonic mean of average precision and average recall, formally defined as:

$$F_\beta = \frac{(1 + \beta^2) \cdot Presicion \cdot Recall}{\beta^2 \cdot Presicion + Recall} \tag{3.10}$$

In our comparison, similar to (Fan et al., 2020) , we set $\beta^2 = 0.3$ to weigh precision more than recall and we calculate the max *F*-measure using various fixed thresholds within the interval of [0, 255].

S-measure: Behavioral vision studies confirm the assertion that the human visual system is extremely sensitive to *S*-measure was developed to assess a generated saliency map by taking into account the structural similarity between a predicted saliency and a GT map. *S*-measure incorporates both the region-aware (S_r) and the object-aware (S_o) structural similarity in a final structure metric, calculated as:

$$S_a = b \cdot S_o + (1 - b) \cdot S_r \tag{3.11}$$

where $b \in [0,1]$ denotes a balance parameter. As in (Fan et al., 2020), we set this parameter to 0.5.

Table 3.3. Comparison of MonoSOD and D^{3}Net

Metrics	MonoSOD		D^{3}Net (Fan et al., 2020)		
	DenseDepth	*Monodpeth2*	*DenseDepth*	*Monodpeth2*	*Prov. Depth*
$F_\beta \uparrow$	0.875	0.869	0.857	0.849	0.896
$E_\xi \uparrow$	0.927	0.924	0.920	0.919	0.947
$S_a \uparrow$	0.893	0.891	0.898	0.895	0.903
$MAE \downarrow$	0.042	0.042	0.043	0.045	0.039

↑means higher is better and vice versa.

Table 3.4. Comparison of MonoSOD with and without the refinement step

Datasets	Metrics	MonoSOD		MonoSOD w/o Ref.	
		DenseDepth	*Monodepth2*	*DenseDepth*	*Monodepth2*
NJU2K (Ju et al., 2014)	$F_\beta \uparrow$	0.877	0.873	0.861	0.860
	$E_\xi \uparrow$	0.917	0.916	0.908	0.908
	$S_a \uparrow$	0.886	0.886	0.892	0.891
	$MAE \downarrow$	0.049	0.048	0.051	0.051
NLPR (H. Peng et al., 2014)	$F_\beta \uparrow$	0.878	0.867	0.855	0.844
	$E_\xi \uparrow$	0.944	0.939	0.939	0.936
	$S_a \uparrow$	0.904	0.899	0.909	0.904
	$MAE \downarrow$	0.030	0.032	0.032	0.034
STERE (Niu et al., 2012)	$F_\beta \uparrow$	0.871	0.868	0.857	0.855
	$E_\xi \uparrow$	0.921	0.919	0.915	0.914
	$S_a \uparrow$	0.889	0.889	0.894	0.893
	$MAE \downarrow$	0.047	0.046	0.050	0.049

↑means higher is better and vice versa.

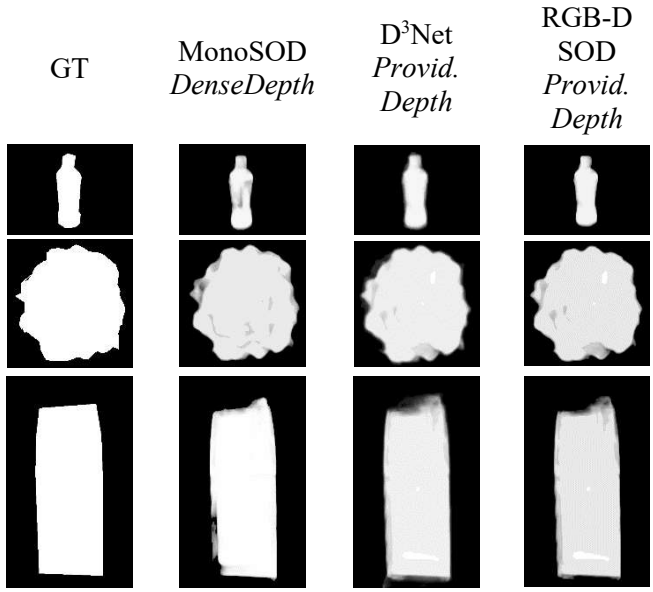
Table 3.5. MonoSOD vs. RGB-D SOD

Metrics	MonoSOD		RGB-D SOD
	DenseDepth	*Monodepth2*	*Prov. Depth*
$F_\beta \uparrow$	0.875	0.869	0.877
$E_\xi \uparrow$	0.927	0.924	0.930
$S_a \uparrow$	0.893	0.891	0.897
$MAE \downarrow$	0.042	0.042	0.040

E-measure: Cognitive vision studies ascertain the fact that human vision is highly sensitive to both global and local information in scenes. Although the most evaluation metrics capture various types of errors in either pixel-wise or structural way, E-measure taking into consideration both pixel-level errors and image-level errors. This metric introduces the enhanced alignment matrix to

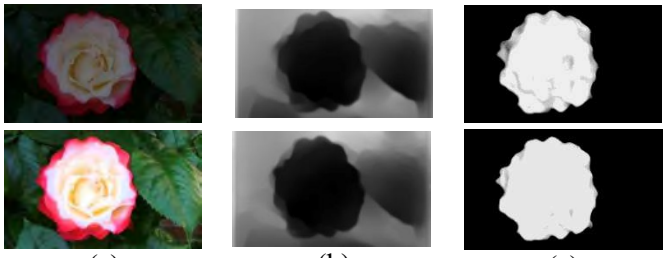
(a) (b) (c)

Figure 3.11. Example of RGB input images depicting salient objects. (a) Bottle. (b) Flower. (c) ATM.



(a) (b) (c) (d)

Figure 3.12. Comparative qualitative results on the images of Fig. 3. (a) GT image. (b) MonoSOD. (c) D^{3}Net (Fan et al., 2020). (d) RGB-D SOD. All images are normalized for better visualization.



(a) (b) (c)

Figure 3.13. (a) Input images of Fig. 3(a) for different luminosities; (b) Corresponding estimated D_M of (a); (c) MonoSOD prediction.

simultaneously formulate local pixel values with the image-level mean value. Thus, it captures both global statistics and local pixel matching information. *E*-measure is calculated as:

$$E_\xi = \frac{1}{w \cdot h} \sum_{x=1}^{w} \sum_{y=1}^{h} \varphi_{F_M}(x, y) \qquad (3.12)$$

where h, w corresponds to the height and the width of the map, respectively, and φ_{F_M} denotes the enhancement alignment matrix of the binary foreground estimated map (F_M).

The performance of the proposed methodology was assessed in comparison to the state-of-the-art for SOD, on all the benchmark datasets, using the aforementioned evaluation metrics. The quantitative results are summarized in Table 3.3. MonoSOD using DenseDepth provided higher scores as compared to MonoDepth2. However, MonoDepth2 follows a self-supervised training approach that it could justify its lower performance. The state-of-the-art SOD model proposed in (Fan et al., 2020) (D3Net) was evaluated on the same datasets as MonoSOD. Also, it has been trained and tested using both the sensor-based estimated depth provided (Provid. Depth) along with the datasets and predicted depth maps. MonoSOD had an overall higher performance than D3Net when both used predicted depth. On the other hand, D3Net achieved marginally higher scores (of the order of ~10-2 for F_β, E_ξ, S_a and of the order of ~10^{-3} for *MAE*) when it was trained with sensor-based estimated depth. It is worth noting that

D^3Net is more complex than MonoSOD, since it uses 3 separate autoencoder-like networks. MonoSOD follows a two-branch architecture capable of predicting depth and subsequently combining it with RGB information for SOD estimation.

The impact of the refinement step to the overall SOD methodology, was evaluated by an additional experiment, where the refinement step was absent from the inference pipeline. The results obtained are summarized in Table 3.4 and indicate that this step can substantially improve the performance of MonoSOD. The MonoSOD architecture, equipped with the refinement process, provided higher scores on most evaluation metrics. Furthermore, to investigate the effect of depth accuracy on SOD performance, MonoSOD was trained with the sensor based estimated depth provided along with the respective datasets. In the following we refer to this model as RGB-D SOD. The results of this comparison are presented in Table 3.5. As it can be observed, RGB-D SOD provides comparable results to MonoSOD, which reinforces the claim that the predicted depth can replace the sensor-based estimated depth.

A qualitative comparison among MonoSOD, D^3Net and RGB-D SOD is illustrated in Figure 3.12 using representative images of the benchmark datasets (Figure 3.11). The GT is provided in Figure 3.12 (a). Figure 3.12 (b), Figure 3.12 (c) and Figure 3.12 (d) present the detected salient objects using MonoSOD, D^3Net and RGB-D SOD, respectively. It can be noticed that the differences between them are marginal. Additionally, in Figure 3.13 is illustrated a qualitative comparison of the performance of MonoSOD using different luminosities. In Figure 3.13 (a) the images in the first and second row illustrate the image of Figure 3.11 (b) with a luminosity of -50% and +50% respectively. It can be observed that the MonoSOD prediction is only slightly affected by the luminosity change.

3.4 Conclusions and Future Work

Towards the development of models able to perceive the salient regions in images, in section 3.1 two different CNN model architectures have been presented designed for saliency estimation on WCE images based on eye fixations of physicians. As distinct from other approaches the proposed models are capable to efficient detect gaze patterns for a variety of normal and abnormal WCE image categories. According to the results, the enhanced architecture that was developed that comprises a convolutional reconstruction block and a post-refinement step outperformed both the basic model and the state-of-the art methods that have been proposed for applications in the same context. An extension to the methodology that was described in section 3.1 has been introduced in section 3.2 where a novel CNN model is presented for estimating saliency maps based on physician's eye fixations on WCE images. The proposed method differentiates from other approaches in the sense that is capable to efficient detect gaze patterns for a variety of normal and abnormal WCE image categories, as well as it utilizes a novel co-operative training scheme. This training scheme is similar to Generative Adversarial Networks but instead of utilizing two networks competing each other, we have incorporated two co-operative models (Goodfellow et al., 2014).

Apart from the work that was presented aiming to tackle the estimation of salient regions, in section 3.3 a novel depth-aware, two-branch CNN salient object detection methodology, called MonoSOD, that unlike relevant state-of-the-art methodologies utilizes predicted depth instead of sensor-based estimated depth is proposed. This method in contrast to salient region prediction, focuses on the detection of salient objects, *i.e.*, separated objects that attract the visual attention. The results provided by the evaluation of the proposed methodology, indicate that the incorporation of depth predicted by a CNN model, can provide comparable performance for SOD, with that of sensor-based SOD methods. MonoSOD outperforms the state-of-the-art D^3Net when both utilize predicted depth, with a smaller architectural complexity (D^3Net has a three-branch architecture, whereas MonoSOD has a two-branch architecture). This work also contributed to the comparison among the performance of different models, given predicted and sensor-based estimated depth. The results lead to the conclusion that there is only a marginal difference, both from a quantitative and qualitative perspective, in terms of SOD performance, between the utilization of predicted and sensor-based estimated depth. MonoSOD can be beneficial for robotic applications where the installation of sensor-based depth acquisition methods is difficult due to the design requirements of the robot, *e.g.*, in robotic capsule endoscopes (Ciuti et al., 2016).

Regarding the salient region estimation, as a future work we intend to investigate different CNN architectures and image representations, *e.g.*, using CIE-Lab color space instead of RGB, for the improvement of the saliency estimation. Considering the results obtained from the study of SOD, as a future work, we intend to further investigate ways for a more robust monocular SOD methodology through application-specific depth representation and approximation methods. Additionally, we intend to further improve the architecture of the model, by reducing its computational complexity with the minimum trade-off in terms of performance. Furthermore, the results obtained by MonoSOD should be a motivation for further investigation to determine which aspect of depth information is beneficial for a network; the pixel-level precision of depth values or a rougher representation of the depth in a scene?

Chapter 4
Obstacle Detection

According to the World Health Organization (WHO), about 16% of the worldwide population lives with some type of visual impairment (WHO, 2018). Visually challenged people (VCP) struggle in their everyday life and have major difficulties in participating in sports, cultural, tourist, family, and other types of outdoor activities. The last two decades, a key solution to this problem has been the development of assistive devices able to help, at least partially, the VCP to adjust in the modern way of life and actively participate in different types of activities. Such assistive devices require the cooperation of researchers from different fields, such as medicine, smart electronics, computer science, and engineering. So far, as a result of this interdisciplinary cooperation, several designs and components of wearable camera-enabled systems for VCP have been proposed (Caraiman et al., 2017; Schwarze et al., 2016; Suresh, Arora, Laha, Gaba, & Bhambri, 2017; Tapu, Mocanu, & Zaharia, 2017).

Such systems incorporate sensors, such as cameras, ultrasonic sensors, laser distance sensors, inertial measurement units, microphones, and GPS, which enable the user identify his/her position in an area of interest (i.e., outdoor environment, hospital, museum, archaeological site, etc.), avoid static or moving obstacles and hazards in close proximity, and provide directions not only for navigation support, but also for personalized guidance in that area. Also, mobile cloud-based applications (Z. Mahmood, Bibi, Usman, Khan, & Muhammad, 2019), methodologies for optimal estimation of trajectories using GPS and other sensors accessible from a mobile device (Ahmed et al., 2019), and algorithms enabling efficient data coding for video streaming (Khan et al., 2019), can be considered for enhanced user experience in this context. Users should be able to easily interact with the system through speech in real-time. Moreover, the system should be able to share the navigation experience of the user not strictly as audio-visual information, but also through social interaction with remote individuals, including people with locomotor disabilities and the elderly.

During the last two years, several studies and research projects have been initiated, setting higher standards towards systems for computer-assisted navigation of VCP. In (Bashiri et al., 2018), an Enterprise Edition of a Google glass device was employed to support visually challenged individuals during their movement. Their system comprised a user interface, a computer network platform, and an electronic device to integrate all components into a single assistive device. In (K. Yang et al., 2017), a commercial pair of smart glasses (KR-VISION), consisting of an RGB-D sensor (RealSense R200) and a set of bone-conducting earphones, was linked to a portable processor. The RealSense R200 sensor was also employed in (Long, Wang, Cheng, Hu, & Yang, 2019), together with a low-power millimetre wave (MMW) radar sensor, in order to unify object detection, recognition, and fusion. Another smart assistive navigation system comprised a smart-glass with a Raspberry Pi camera attached on a Raspberry Pi processor, as well as a smart shoe with an IR sensor for obstacle detection attached on an Arduino board (Pardasani, Indi, Banerjee, Kamal, & Garg, 2019). In (B. Jiang, Yang, Lv, & Song, 2019), a binocular vision probe with two charged coupled device (CCD) cameras and a semiconductor laser was employed to capture images in a fixed frequency. A composite head-mounted wearable system with a camera, ultrasonic sensor, IR sensor, button controller, and battery for image recognition was proposed in (S. Chen, Yao, Cao, & Shen, 2019). Two less complex approaches were proposed in (Adegoke, Oyeleke, Mahmud, Ajoje, & Thomase, 2019) and (M. T. Islam, Ahmad, & Bappy, 2020). In the first, two ultrasonic sensors, two vibrating motors, two transistors, and an Arduino Pro Mini Chip were attached on a simple pair of glasses. The directions were provided to the user through vibrations. In the second, a Raspberry Pi camera and two ultrasonic sensors attached on a Raspberry Pi processor were placed on a plexiglass frame.

The previously mentioned systems incorporate several types of sensors, which increase the computational demands and the energy consumption, the weight of the wearable device, as well as the complexity of the system. In addition, although directions in the form of vibrations are faster perceivable, their expressiveness is limited, and the learning curve required increases with the number of messages needed for user navigation.

From the review of the available systems above, it can be easily concluded that the commercially available assistive technologies for navigation are mainly based on Global Positioning System (GPS). GPS assistive guidance approaches are unsuitable for VCPs since they lack high accuracy in urban environments (the error can be estimated even to several meters); the signal can be easily lost during operation for various reasons such as line-of-sight restrictions or multi-path effect. In addition, GPS needs sufficient number of directly visible satellites to work properly (Rodrıguez, Bergasa, Alcantarilla, Yebes, & Cela, 2012).

A factor that would make an assistive system for navigation suitable for VCPs is its ability to detect various types of objects in images by computer vision. The analysis of user requirements of the project ENORASI (D.K. Iakovidis, 2019), showed that the users would like the system to detect in real-time mainly vertical objects such as trees, humans, stairs and terrain anomalies, while in parallel the system should be able to accurately and reliably detect the surrounding area for

potential cultural sights. Computer vision methods for obstacle detection are also important in the robotic navigation domain (Brassai, Iantovics, & Enachescu, 2012), so any methodology being developed for a form of assistive/independent navigation can be applied for the navigation of VCPs to the robotic navigation domain and vice-versa.

Several studies investigating deep learning approaches have addressed the issue of object detection in images. A Faster Region-Based Convolutional Neural Network (R-CNN) (Ren, He, Girshick, & Sun, 2015) for real-time object detection with region proposal networks was used to detect and track objects in (Kaur & Bhattacharya, 2018). Motion, sharpening and blurring filters were used to enhance feature representation. An approach enabling joint object detection, tracking and recognition was developed in the context of the DEEP-SEE framework, presented in (Tapu et al., 2017). An intelligent smart glass using deep learning machine vision techniques and Robot Operating System (ROS) was presented in (Suresh et al., 2017), where three CNN architectures were used, namely Faster R-CNNs (Ren et al., 2015), You Only Look Once (YOLO) CNN (Redmon, Divvala, Girshick, & Farhadi, 2016) and Single Shot multibox Detectors (SSDs) (W. Liu et al., 2016). However, the main purpose of these methods was to solely detect objects and not classify them as obstacles.

An obstacle detection module of a wearable mobility aid based on LeNet was proposed in (Poggi & Mattoccia, 2016), and a unified real-time object detection method based on a YOLO CNN was proposed in (Redmon et al., 2016). These machine learning-based methods consider obstacle detection as a 2D problem in the image plane. A few studies have considered obstacle detection for VCP as a 3D problem by incorporating depth information. Such information is usually derived from stereo vision systems (see section 2.1.2). A multi-stage depth-aware random forest model using discriminative saliency fusion was developed for salient region detection (H. Song, Liu, Du, & Sun, 2016). For micro air vehicle flight applications, a multi-task deep architecture that jointly estimates depth and obstacles without computing a global map was proposed (Mancini, Costante, Valigi, & Ciarfuglia, 2018). For traffic situations, a stereo and motion fusion approach using flow/depth constraint for obstacle detection was developed in the context of the Daimler-Chrysler urban traffic assistance project (Heinrich, 2002).

A key component of all these systems and methodologies that aim at tackling the problem of assistive navigation is that they exploit information deriving from multiple sensors or complex sensory systems such as stereo vision cameras. Under the scope of sensory substitution several systems have been developed integrating electronic devices to enhance the user's cognition regarding the surroundings. This is achieved through the incorporation of various sensors such as ultrasonic (Bottega & Balbinot, 2020; Rahman, Islam, Ahmmed, & Khan, 2020) and specialized camera systems capable of providing both RGB and depth information that describe a scene (Barontini, Catalano, Pallottino, Leporini, & Bianchi, 2020).

More recently, various deep learning approaches have been proposed to comprehensively capture the surrounding environment. Joshi *et al.* (Joshi, Yadav, Dutta, & Travieso-Gonzalez, 2020) proposed a navigation system leveraging deep learning-based object detection, that utilizes

distance sensors to assist VIPs through audio feedback. Similarly, in (M. M. Islam, Sadi, & Bräunl, 2020; Shahira, Tripathy, & Lijiya, 2019; Yadav, Joshi, Dutta, Kiac, & Sikora, 2020) ultrasonic sensors were used alongside deep-learning models for obstacle detection. In (Afif, Ayachi, Said, Pissaloux, & Atri, 2020; Hsieh, Lin, & Xu, 2020; O.-C. A. Liu, Li, Yan, Ng, & Kwok, 2020), Convolutional Neural Network (CNN) models were employed for object detection and recognition in indoor spaces for the assistive navigation of the VCPs.

With the exploitation of the segmentation capabilities of deep learning methods, several systems have been developed to guide the visually impaired (Duh, Sung, Chiang, Chang, & Chen, 2020), (Hengle, Kulkarni, Bavadekar, Kulkarni, & Udyawar, 2020). Bauer *et al.* (Bauer et al., 2020) presented a wearable device that exploits the 3D information of the environment to pinpoint the relative locations of objects in the surrounding area. Additionally, the detected objects were semantically described, to convey them to the user as audio or haptic feedback. Leveraging the depth information acquired from an RGB-D camera whilst performing segmentation, the methodologies presented in (Haoye Chen et al., 2020; Martinez, Yang, Constantinescu, & Stiefelhagen, 2020) were proposed for the guidance of VIPs. All these approaches, however, are segmenting a scene to semantically isolate spatial regions, while assessing their risk using hard thresholds on depth information estimated by additional sensors. Some works have considered the application of self-supervised learning techniques to perform obstacle detection under differing settings, including robotic navigation (X. Deng et al., 2020; Kahn, Abbeel, & Levine, 2021), depth estimation for vision-based vehicles (Tektonidis & Monnin, 2020) and obstacle avoidance for unmanned aerial vehicles (Dijk, 2020). However, only a few self-supervised methodologies indirectly address the assistive navigation for VIPs, by discovering terrain characteristics (Ishikawa, Hachiuma, & Saito, 2021; Kurobe, Nakajima, Saito, & Kitani, 2020) and generating natural language image captions (Y. Song, Chen, Zhao, & Jin, 2019).

By considering all the above, in this dissertation various methodologies that go beyond the state-of-the-art regarding assistive navigation have been investigated. In detail, three methodologies based on DL and Fuzzy Logic have been developed aiming at tackling the problems of obstacle detection and avoidance. The first method that is presented in section 4.1 introduces a novel direction for tackling the problem of obstacle detection based on a Generative Adversarial Network (GAN) trained to detect salient regions on images and fusing them with spatial risk assessed using depth maps estimated by an RGB-D sensor. The fusion between the saliency and risk information is performed with the use of fuzzy sets. Unlike previous approaches the proposed method enables the assessment of the degree to which an obstacle imposes risk to the user, and it provides an inherent way of naturally describing the current situation to the user using linguistic values. The use of fuzzy sets enables soft assessment of the bounds of a threat (*e.g.*, there are overlapping distance intervals within which an obstacle can be considered both as a high or medium risk threats) which is proved to be more effective than conventional approaches, usually employing hard bounds (Long Chen, Guo, & Sun, 2010), *e.g.,* an obstacle is considered as a threat of high risk after a specified distance. The human eye fixation map produced by the GAN, combined with the fuzzy interpretation of the depth values, produce a perceptually meaningful and interpretable

information for the user. Additionally, this method when compared with the machine learning-based obstacle detection methods that can be found in the literature, does not need any training regarding the obstacle detection part.

Then, in section 4.2 an extension of the first methodology is presented that incorporates additional modules such as personalization elements, a simple ground removal approach for the elimination of false positive findings and a linguistic encoding of the position of the obstacle to communicate commands for their avoidance. In section 4.4 a novel deep learning-based obstacle detection methodology is investigated that applies self-supervised learning. It is based on a CNN model that is trained with input and output pairs of a state-of-the-art obstacle detection algorithm that is described in section 4.2, aiming to approximate it through simulation. Unlike the methodologies presented in sections 4.1 and 4.2, it uses only the RGB components of the input images (*i.e.*, the depth component is excluded), and it is significantly less computationally intensive. The model learns to infer a fuzzy saliency map softly representing image regions corresponding to obstacles in the environment, considered of high-risk for possible collision of the subject that is bearing the camera. According to the methodology described in 4.2, the saliency maps are subsequently utilized for obstacle localization using bounding boxes. The self-supervised aspect of this methodology emerges from the use of automatically generated training samples produced by the methodology described in 4.2. Thus, in contrast to other methods that either predict depth (Tektonidis & Monnin, 2020) or involve additional sensors, such as ultrasound sensors (M. M. Islam et al., 2020; Shahira et al., 2019; Yadav et al., 2020), the model that we employ, implicitly learns all the subprocess of an obstacle detection method (section 4.2) given a single RGB input image. The motivation for this research lies on the fact that currently, obstacle detection algorithms require the incorporation of multiple sensors and computationally expensive methods for the assessment of obstacles in images. These approaches affect the portability and the response time of the systems designed for assistive navigation. A single CNN model, that can efficiently detect regions with obstacles in an image, would contribute to less complex and more time-efficient navigational systems.

4.1 Soft Obstacle Detection Using GANs and Fuzzy Logic

Figure 4.1. Illustration of the SalGAN architecture of the generator.

The proposed method consists of two components; the saliency map generation using a GAN trained on human eye fixations, called SalGAN (J. Pan et al., 2017) and a fuzzy set-based approach combining the 3D spatial information acquired by an RGB-D sensor to risk values for a possible obstacle threat.

The saliency map generation is based on a GAN (Goodfellow et al., 2014), which is a deep CNN (Fig. 1). The main motivation utilizing it in the saliency prediction, is that it produces saliency maps based on human eye fixations. Thus, an eye fixation-based salient map carries the information of what regions in an image would be interesting for a human. So forth, the obstacle detection process can become more intuitive, in the sense that the algorithm will be able to detect regions that humans consider as more salient. Furthermore, it can be trained with eye fixation data captured from individuals navigating through paths with obstacles, and hence extend its potentials.

(a) (b)

Figure 4.2. Illustration of the generated saliency map from an input image. (a) Input image. (b) The generated saliency map

The GAN architecture consists of two CNNs, a discriminator and a generator network. The combination of the two CNNs aims to predict visual saliency maps from an image. The generator produces the saliency maps in its output, and the discriminator optimizes its resemblance with ground truth saliency maps obtained from humans using eye tracking data (Bylinskii et al., 2016). The generator CNN has an encoder-decoder architecture. The encoder part has the same architecture as VGG-16 (Simonyan & Zisserman, 2014), including both convolutional and pooling layers (illustrated in grey and purple colour in Figure 4.1. However, it does not have the final pooling and fully connected layers (FC) layers of VGG-16. The max-pooling layers of the encoder are used for downscaling of the feature maps. The encoder network weights are initialized with those of a VGG-16 model trained on the ImageNet dataset (J. Deng et al., 2009). For the purpose of estimating saliency maps, only the last two groups of convolutional layers have been modified

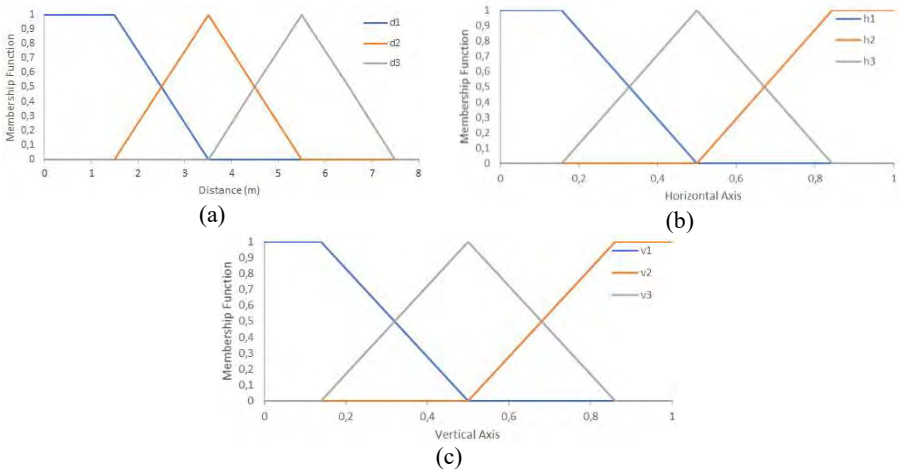


Figure 4.3. Membership functions of fuzzy sets used for the localization of objects in the 3D space using linguistic variables. (a) Membership functions for low (d_1), medium (d_1), and high risk (d_3) upon the distance of the user from an obstacle. (b) Membership functions for left (h_1), central (h_2) and right (h_3) positions on the horizontal axis. (c) Membership functions for up (v_1), central (v_2) and bottom (v_1) positions on the vertical axis.

during training. The decoder part of the CNN has up-sampling instead of pooling layers followed by convolutional filters in order to increase the size of the output to match the size of the input image. The decoder architecture is identical to the encoder architecture but with the layers placed in reverse order. as an activation function, Rectified Linear Unit (ReLU) was used in all convolutional layers. To produce the saliency map, a final 1×1 convolutional layer was places with a sigmoid activation function. The weights of the decoder were randomly initialized. The discriminator CNN architecture consists of six 3×3 convolution filters with 3 pooling layers and followed by 3 FC layers. The activation functions for the convolutional and FC layers are the ReLU and hyperbolic tangent tanh function, respectively. Exception is the final layer, which uses the sigmoid activation function. An example of a saliency map produced by the generative model, is illustrated in Figure 4.2. In Figure 4.1 is illustrated the architecture of the generative model. The convolutional layers of the encoder are depicted with a blue colour and the max-pooling layers with red; the convolutional layers of the decoder are depicted with green colour, the up-sampling layers with orange and th e final convolutional layer with the sigmoid activation with gray colour.

The generated saliency map can be seen as a broad weighted region of interest in which an obstacle may reside, *i.e.*, higher intensities of the saliency map correspond to higher likelihoods for possible objects of interest. To limit the search region for the detection of an obstacle, we exploit the values of a depth map produced by an RGB-D sensor. To assess the risk of a threatening obstacle for the VII, we employ a methodology based on fuzzy sets (Nguyen, Walker, & Walker, 2018). We consider a set of 3 fuzzy sets Δ_1, Δ_2, Δ_3, which correspond to three different risk levels, expressible by linguistic values indicating high, medium, and low risk respectively. Each fuzzy set represents a degree of risk that an object in each depth value z, may impose a threat to the VII. The universe of discourse for these fuzzy sets is the range of values of the depth maps produced by the RGB-D sensor. The fuzzy sets Δ_1 and Δ_2, and the fuzzy sets Δ_2 and Δ_3, are overlapping between each other,

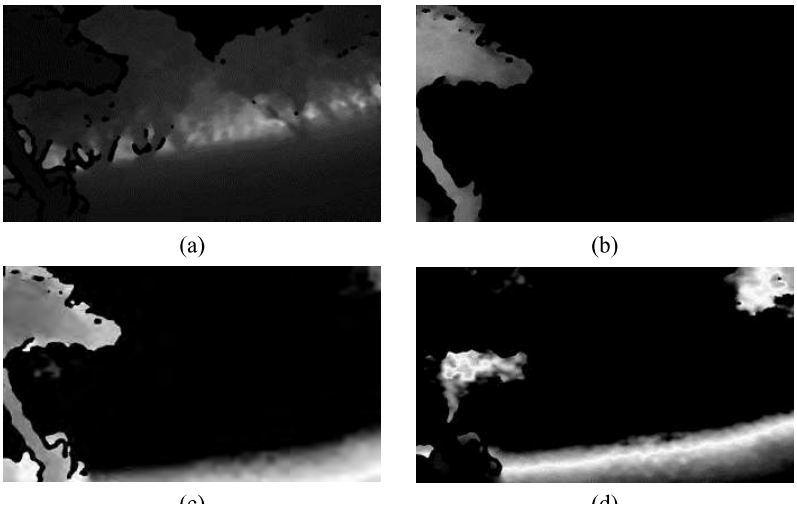


Figure 4.4. A graphical representation of the risk maps. (a) Depth map M_z of the image in **Figure 4.2**(a). (b) High-risk map $\boldsymbol{\Delta_M^1}$. (c) Medium-risk map $\boldsymbol{\Delta_M^2}$. (d) Low-risk map $\boldsymbol{\Delta_M^3}$.

considering the uncertainty in the assessment of an obstacle threat as high or medium, and as medium or low respectively, upon its distance from the user. The respective membership functions $d_i(z)$, i = 1, 2, 3 are illustrated in Figure 4.3(a), where $z \in [0, \infty)$ is a value of the estimated depth map.

For the spatial localization of an obstacle in the image plane, we constructed 6 additional fuzzy sets, namely H_1, H_2 and H_3 for the horizontal axis, corresponding to the left, central and right part of the image, namely, V_1, V_2 and V_3 for the vertical axis, corresponding to the up, central, and bottom part of the image. The respective membership functions, $h_i(x)$, and $v_i(y)$ are illustrated in Figure 4.3 (b), (c), where $x \in [0, 1]$ and $y \in [0, 1]$ are the horizontal and vertical coordinates within an image, normalized by image width and height, respectively.

Once we establish the membership functions for each fuzzy set, we create three different risk maps, Δ_M^i based on the depth values and the membership functions of each fuzzy set Δ_i. Each risk map consists of the responses of a membership function given a depth value z and it can be formally expressed as follows:

$$\Delta_M^i(x, y) = d_i\big(M_z(x, y)\big) \tag{4.1}$$

where M_z is the depth map of an RGB image, I_{RGB}. From Eq. (4.2) a total of 3 risk maps is derived, where each, represents regions of a degree of risk that an object may impose a threat to a person navigating within its range. A visual representation of the risk maps can be seen in Figure 4.4. Figure 4.4 (a) illustrates the depth map corresponding to Figure 4.2 (a), where the dark pixel values represent lower depth values (nearest distances) and the brighter pixel values represent higher depth values (further distances). Figure 4.4 (b)-(d) are illustrations of the different risk maps produced by Eq.(4.2) using membership functions $d_i(z)$, i=1, 2, 3. The brighter pixel values of the risk maps indicate higher participation in Δ_1, Δ_2 and Δ_3 fuzzy sets respectively.

To further localize an obstacle imposing threat to a navigating VII, we combine the information from the saliency map S_M with a risk map Δ_M^i, using the fuzzy AND operator (∧) (Nguyen et al., 2018) :

$$F_1 \wedge F_2 \coloneqq \min\big(F_1(x, y), F_2(x, y)\big) \tag{4.2}$$

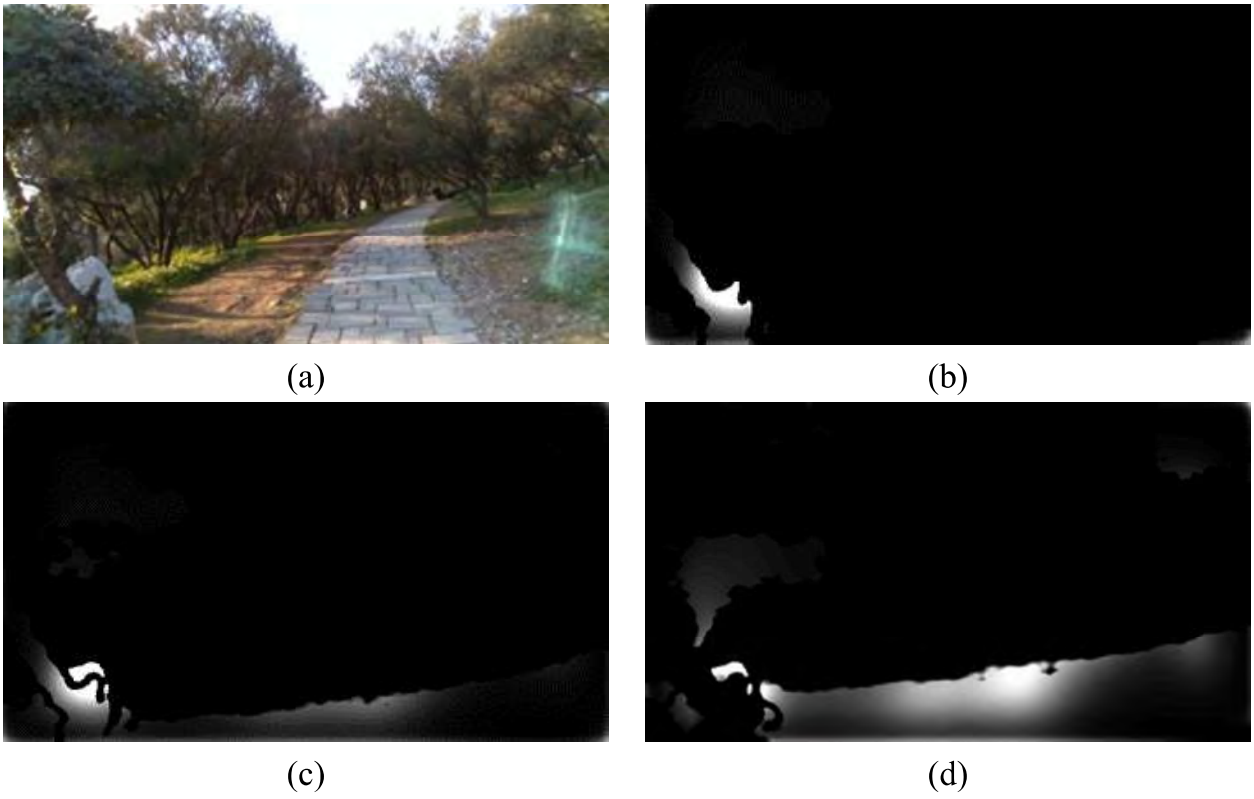


Figure 4.6. A visual representation of the $\boldsymbol{O^i_{obstacle}}$. (a) The original IRGB image; (b) Image $\boldsymbol{O^1_{obstacle}}$; (c) Image $\boldsymbol{O^2_{obstacle}}$; (d) Image $\boldsymbol{O^3_{obstacle}}$.

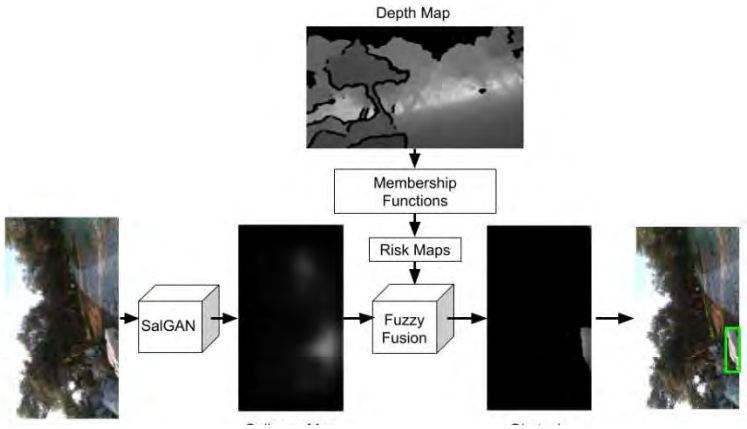


Figure 4.5. A visual representation of the pipeline of the proposed methodology.

where F_1, F_2 are two 2D fuzzy maps, with values within [0, 1], and x, y represents the coordinates of each value within the 2D map. A risk map Δ^i_M and a (normalized) saliency map S_M, can be considered as such fuzzy maps, since the risk maps are generated by the responses of fuzzy membership functions, and saliency maps indicate the degree to which a pixel belongs to a salient region. Thus, Eq.(4.3) gives a new image:

$$O^i_{obstacle} = S_M \wedge \Delta^i_M \qquad (4.3)$$

where non-zero pixel-values of $O^i_{obstacle}$ represent the location of an obstacle and a degree of participation in a risk-level in the respective region. An example of the application of Eq. (4.4) for

the combination of the saliency map of Figure 4.2 with each of the different risk maps of Figure 4.3 is illustrated in Figure 4.6. In Figure 4.6 (b) the whitish area indicates that the respective object in image Figure 4.6 (a) (the tree on the left) is of high-risk so it is labeled as an obstacle to be avoided. The whitish area in Figure 4.6 (c) indicates the objects of medium-risk and the whitish area in Figure 4.6 (d) indicates that the respective objects are of low-risk. The fact that the lower part of the tree is highlighted in both Figure 4.6 (b) and Figure 4.6 (c) is due to the overlap of the respective membership functions. The overall pipeline of the proposed methodology is summarized in Figure 4.7.

Experiments and Results

(a) (b)

Figure 4.7. Example detection of high-risk obstacles (nearest crowd and tree branches). (a) Original image IRGB. (b) The corresponding image $\boldsymbol{O}^{\boldsymbol{1}}_{\boldsymbol{obstacle}}$.obtained by using membership function d_1.

For the validation of our method, we captured 10,170 frames in total. The videos were captured using an RGB-D sensor, namely, the state-of-the-art Intel® RealSense™ D435. The size of this sensor is conveniently small (90×25×25mm) to be attached on a wearable system for the assistive navigation of the VII. It enables 3D depth sensing, with a maximum range of 10m. The D435 sensor has two infrared (IR) cameras that enable stereoscopic vision, an IR projector and a high resolution RGB camera. The IR projector is used to improve the depth estimation. This is done by the stereoscopic system while projecting a static IR pattern on the scene. The IR pattern projection enables the texture enrichment of low texture scenes.

Table 4.1. Percentage confusion matrix

Predicted Values	True Values	
	Positive	*Negative*
Positive	47.50%	6.66%
Negative	5.20%	40.62%

The originally captured videos were uniformly subsampled, by acquiring a sample every 5 frames. After this process, the resulting dataset was composed of a total of 2034 frames, including 6 obstacle categories, namely trees or tree branches, ground anomalies, crowds and stones. The obstacles in the dataset were annotated by a human observer. For the obstacle detection, we used the high-risk map Δ^1_M.

(a) (b) (d)

Figure 4.8 An illustration of different fuzzy operation between the saliency map and the risk map $\boldsymbol{\Delta_M^1}$. (a) Fuzzy AND. (b) Fuzzy Sum. (c) Fuzzy OR.

The interval of the high-risk membership function was set to be at $0 < z \leq 3.5$m. The interval was decided from the user requirements described in (Dimitris K Iakovidis et al., 2020) . The desired distance for an obstacle to be detected, according to the VII, was up to 2m. We can interpret this statement as follows: "an obstacle is of high-risk to a VII when it is within a range of 2m". Regarding this interpretation, we can assume that all the obstacles within a range of 2m are of high risk but after the 2m threshold is safe. Unlike previous methodologies, we do not consider this distance as a hard threshold for the identification of an obstacle as threatening. As it can be seen in Figure 4.3 (a) our high-risk membership function is 1 within the interval $0 < z \leq 1.5$m, and it starts degrading up to the 3.5m where it becomes zero. In that manner, we are able to capture the desired distance up to 2m and mark it as high risk but also consider the uncertainty around it, i.e., within the interval $1.5\text{m} < z \leq 3.5\text{m}$. Within the aforementioned interval, the distance values z begin to express a participation in the medium risk domain. This was decided upon the usual practice of the membership being intersected on their middle. This way the risk of an approaching obstacle can be progressively assessed with respect to its distance. In addition, we fulfill the requirements of the VII since the impulses of the high-risk membership function within the interval of $0 < \text{z} \leq 2\text{m}$ are belonging to the Δ_1 fuzzy set with high participation compared to other fuzzy set. This can be clearly seen in Figure 4.3.

The application of the proposed methodology for obstacle detection on the available dataset resulted in an accuracy of 88.1%, with the respective specificity to be 85.9% and the sensitivity 90.1%. A confusion matrix with the detailed results in terms of percentages can be seen in Table 4.1. We compared our approach with a state-of-the-art methodology, where hard thresholding is employed. For a threshold at 3m the hard thresholding method produced an accuracy of 81.1%, with a specificity of 79.1% and a sensitivity of 82.9%, whereas by further reducing the threshold, the detection performance was degraded. A qualitative comparison of the two approaches is illustrated in Figure 4.8. Figure 4.8 (b) illustrates the hard-thresholded saliency map overlaid to the input image. It can be noticed that the rock at the bottom right is not included in the region of interest defined by the saliency map. This region of interest is used to isolate a respective region of the depth map, which is subsequently hard-thresholded to obtain possibly threatening obstacle regions. The result of this process is illustrated in Figure 4.8 (d), where the rock is falsely not considered as a threat. Comparatively in Figure 4.8 (c) the image $O_{obstacle}^1$ obtained by the proposed methodology includes the rock. This is because of the fuzzy fusion applied between the

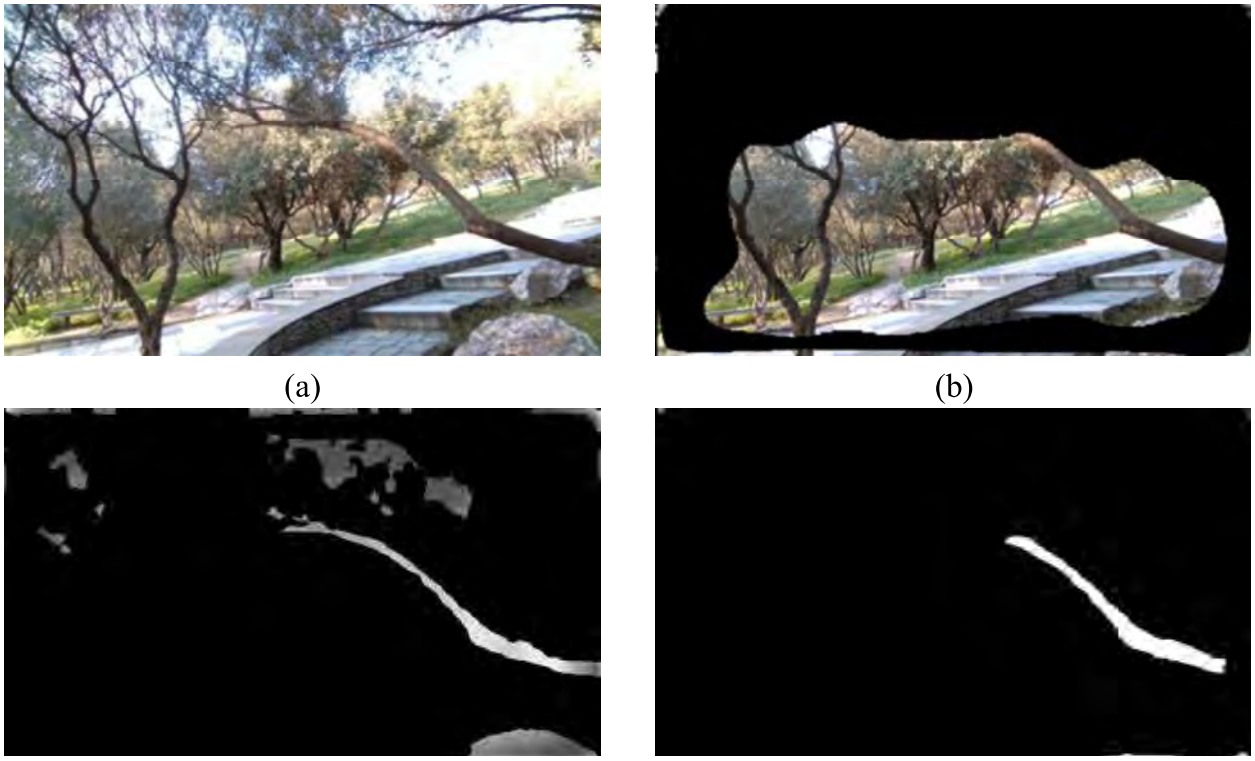


Figure 4.9 A qualitative comparison of the fuzzy approach (a) RGB input image. (b) The hard-thresholded saliency map overlaid to the input image (c) Image $\boldsymbol{O}^{\boldsymbol{1}}_{\boldsymbol{obstacle}}$ obtained by the proposed methodology. (d) Obstacle mask after hard thresholding of the depth map corresponding to the region of interest defined in (b).

saliency map and the risk-map. The fuzzy fusion operation defines a region that may not be considered sufficiently salient by SalGAN, but due to the uncertainty-aware evaluation of the depth map using fuzzy sets, a certain degree of risk is assigned. In Figure 4.9, is illustrated a qualitative comparison among the fuzzy AND (Figure 4.9 (a)), OR (Figure 4.9 (b)) and SUM (Figure 4.9 (c)) operation. In can be observed that the fuzzy AND operation produces the most reliable results.

4.2 Personalized Soft Obstacle Detection

In section 4.1 an obstacle detection methodology has been described that is based on the fusion of the information deriving from estimated human eye-fixations, in terms of saliency maps, and information regarding the risk that is imposed to a user as it is assessed by the depth information that is captured by an RGB-D sensor. In this section, that methodology (section 4.1) is extended with personalization elements and modules that minimize the detection of false positive findings, *i.e.*, characterization of objects as obstacles when in reality impose no harm to the user. This methodology has been merged with other components that lead to the construction of a more complete system for the assistive navigation of the VCPs. However, the main focus of this section

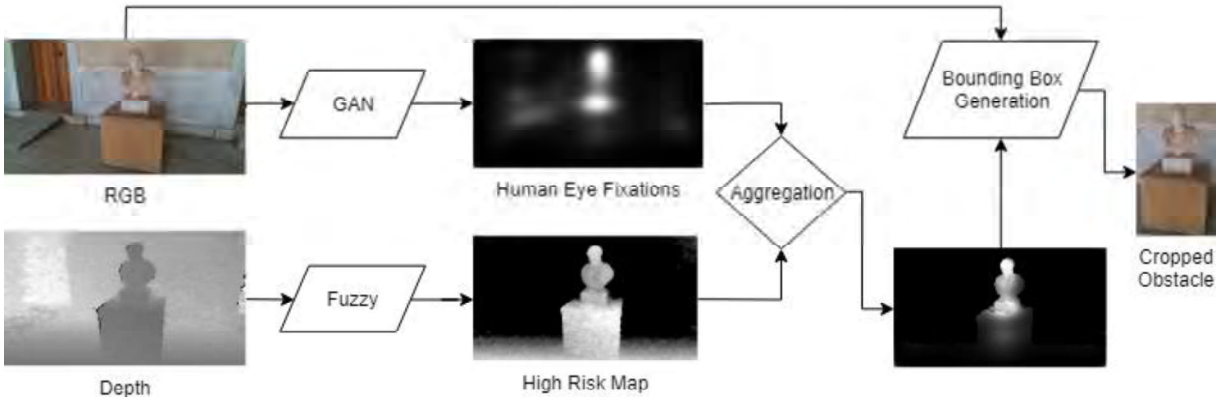


Figure 4.10. Visualization of the proposed obstacle detection and recognition pipeline

will be the obstacle detection and avoidance part of the system whereas the other components, *i.e.*, object recognition, will only be mentioned and not analysed. The obstacle detection and recognition component of the system can be described as a two-step process. In the first step, the detection function incorporates a deep learning model and a risk assessment approach using fuzzy sets similarly to the methodology described in section 4.1. Henceforth, the deep learning model is used to predict, eye-human fixations, on images captured during the navigation of the VCP and sequentially fuzzy sets are used to assess the risk based on depth values calculated by the RGB-D camera, generating risk maps, expressing different degrees of risk. The risk and saliency maps are then fused using a fuzzy aggregation process through which, the probable obstacles are detected. In the second step, the recognition of the probable obstacles takes place. For this purpose, each obstacle region is propagated to a deep learning model, which is trained to infer class labels for objects found in the navigation scenery (Figure 4.10).

The saliency maps used are generated by a GAN (Goodfellow et al., 2014) that have been utilized for the same purpose in section 4.1. The generated saliency maps derive from human eye fixation points and thus, they make the significance of a region in a scene more instinctual. Such information can be exploited for the obstacle detection procedure, and at the same time, enhance the intuition of the methodology. Additionally, the machine learning aspect enables the extensibility of the methodology, since it can be trained with additional eye fixation data, collected from individuals during their navigation through rough terrains. An example of the saliency maps estimated from a given image can be seen in Figure 4.11. Since the model is trained on human eye-fixation data, it identifies as salient, regions in the image that the attention of a human would be focused. As it can be observed in Figure 4.11, in the first image, the most salient region corresponds to the fire extinguisher cabinet; in the second image, to the people on the left side; and in the last image, to the elevated ground and the tree branch.

To assess the risk, it can be easily deduced that objects/areas that are close to the VCP navigating in an area and are salient according to the human gaze, may pose a certain degree of threat to the VCP. Therefore, as a first step, the regions that are in a certain range from the navigating person need to be extracted, so that they can be determined as threatening. Hence, we consider a set of 3

(a) (b)

Figure 4.11. Examples of the generated saliency maps given an RGB image. (a) Input RGB images. (b) respective generated saliency maps.

fuzzy sets, namely, R_1, R_2, and R_3 describing three different risk levels, which can be described with the linguistic values of high, medium, and low risk, respectively. The fuzzy sets R_1, R_2, and R_3, represent a different degree of risk and their universe of discourse is the range of depth values of a depth map. Regarding the fuzzy aspect of these sets and taking into consideration the uncertainty in the risk assessment, there is an overlap between the fuzzy sets describing low and medium, and medium and high risk. The fuzzy sets R_1, R_2, and R_3 are described by the membership function $r_i(z)$, $i = 1, 2, 3$, where $z \in [0, \infty)$. The membership functions are illustrated in Figure 4.12.

A major aspect of an obstacle detection methodology is the localization of obstacles and the description of their position in a manner that it can be communicated and easily perceived by the user. In our system, the description of the spatial location of an object is performed using linguistic

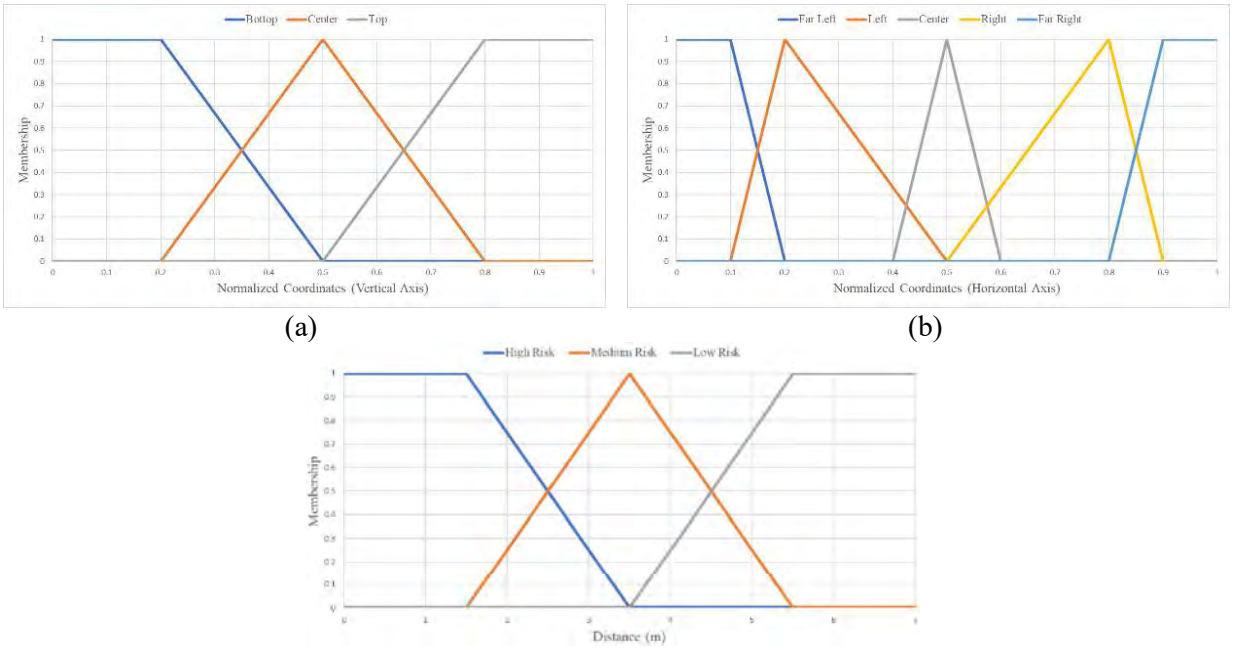


Figure 4.12. Membership functions of fuzzy sets used for the localization of objects in the 3D space using linguistic variables. (a) Membership functions for far left (h_1), left (h_2), central (h_3), right (h_4) and far right (h_5) positions on the horizontal axis. (b) Membership functions for up (v_1), central (v_2) and bottom (v_3) positions on the vertical axis. (c) Membership functions for low (r_1), medium (r_1), and high risk (r_3) upon the distance of the user from an obstacle

expressions. We propose an approach based on fuzzy logic to interpret the obstacle position using linguistic expressions (linguistic values) represented by fuzzy sets. Spatial localization of an obstacle in an image can be achieved by defining 8 additional fuzzy sets. More specifically, we define 5 fuzzy sets for the localization along the horizontal axis of the image, namely, H_1, H_2, H_3, H_4, and H_5 corresponding to far left, left, central, right, and far right portions of the image. Additionally, to express the location of the obstacle along the vertical axis of the image, we define 3 fuzzy sets, namely, V_1, V_2, and V_3 denoting the upper, central, and bottom portions of the image. The respective membership functions of these fuzzy sets are $h_j(x)$, $j = 1, 2, 3, 4, 5$ and $v_i(y)$, $i = 1,2, 3$, where x, $y \in [0, 1]$ are normalized image coordinates. An illustration of these membership functions can be seen in Figure 4.12.

Some obstacles, such as tree branches, may be near the individual with respect to the depth, but at a certain height that safe passage would not be affected. Thus, a personalization step was introduced to the methodology eliminating false alarms. The personalization aspect and the minimization of false positive obstacle detection instances are implemented through an additional fuzzy set P, addressing the risk an obstacle poses to a person with respect to the height. For the description of this P fuzzy set we define a two dimensional membership function $p(h_o, h_u)$, where h_o and h_u are the heights of the obstacle and the user, respectively. More details concerning the personalization aspect of this method will described later.For the risk assessment, since the membership functions describing each fuzzy set were defined, the next step is the creation of 3

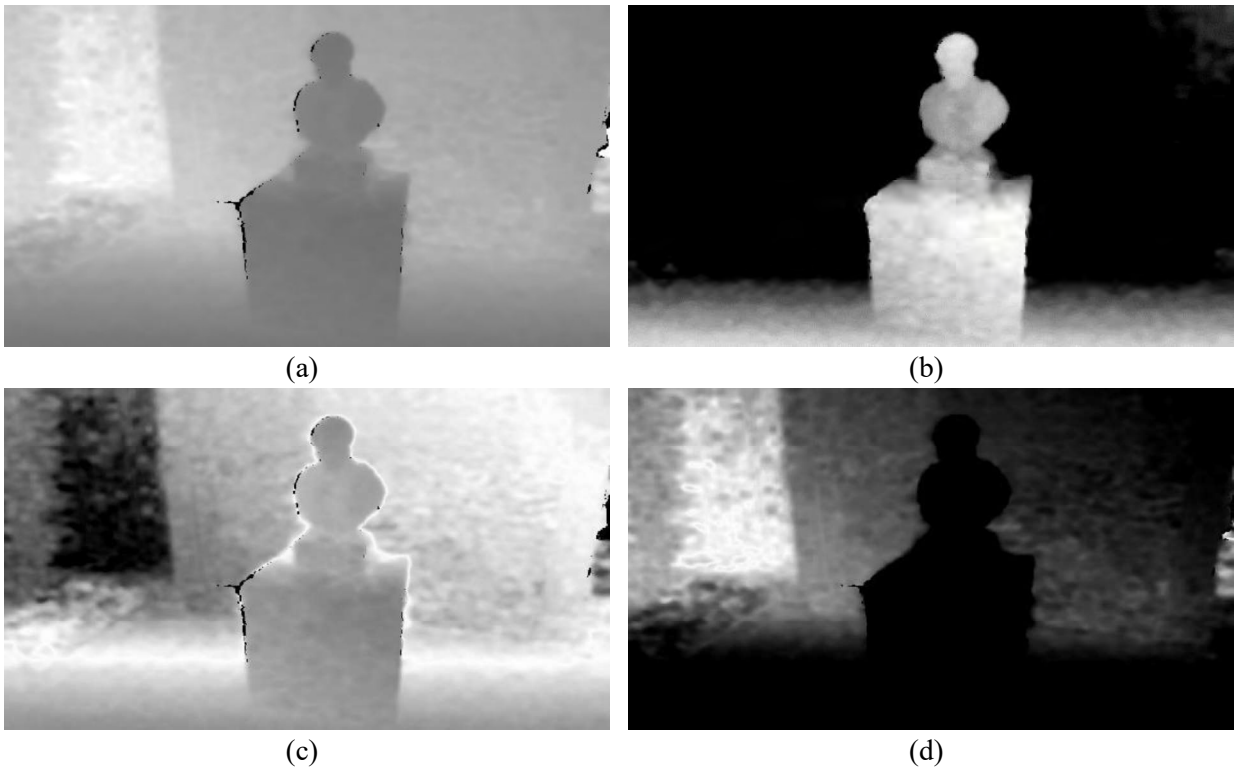


Figure 4.13. Example of $\boldsymbol{R_M^i}$ creation. (a) Depth map *D,* where lower intensities correspond to closer distances; (b) visual representation of $\boldsymbol{R_M^1}$ representing regions of high risk; (c) $\boldsymbol{R_M^2}$ representing regions of medium risk; (d) $\boldsymbol{R_M^3}$ depicting regions of low risk. Higher intensities in (b-d) correspond to lower participation in the respective fuzzy set. All images have been normalized for better visualization.

risk maps, R_M^i. The risk maps R_M^i, derive from the responses of a membership function, $r_i(z)$, and are formally expressed as:

$$R_M^i(x, y) = r_i(D(x, y)) \tag{4.4}$$

where D is a depth map that corresponds to an RGB image IRGB. Using all the risk assessment membership functions, namely r_1, r_2, and r_3, 3 different risk maps, R_M^1, R_M^2, and R_M^3 are derived. Each of these risk maps depicts regions that may pose different degrees of risk to the VCP navigating in the area. In detail, risk map R_M^1 represents regions that may pose high degree of risk, R_M^2 medium degree of risk, and finally R_M^3 low degree of risk. A visual representation of these maps can be seen in Figure 4.13. Figure 4.13 7(b-c) illustrate the risk maps derived from the responses of the r_1, r_2, and r_3 membership functions on the depth map of Figure 4.13 (a). Brighter pixel intensities represent higher participation in the respective fuzzy set, while darker pixel intensities represent lower participation.

In the proposed methodology, the obstacle detection is a combination between the risk assessed from the depth maps and the degree of saliency that is obtained from the GAN described previously. The saliency map S_M that is produced from a given I_{RGB} is aggregated with each risk

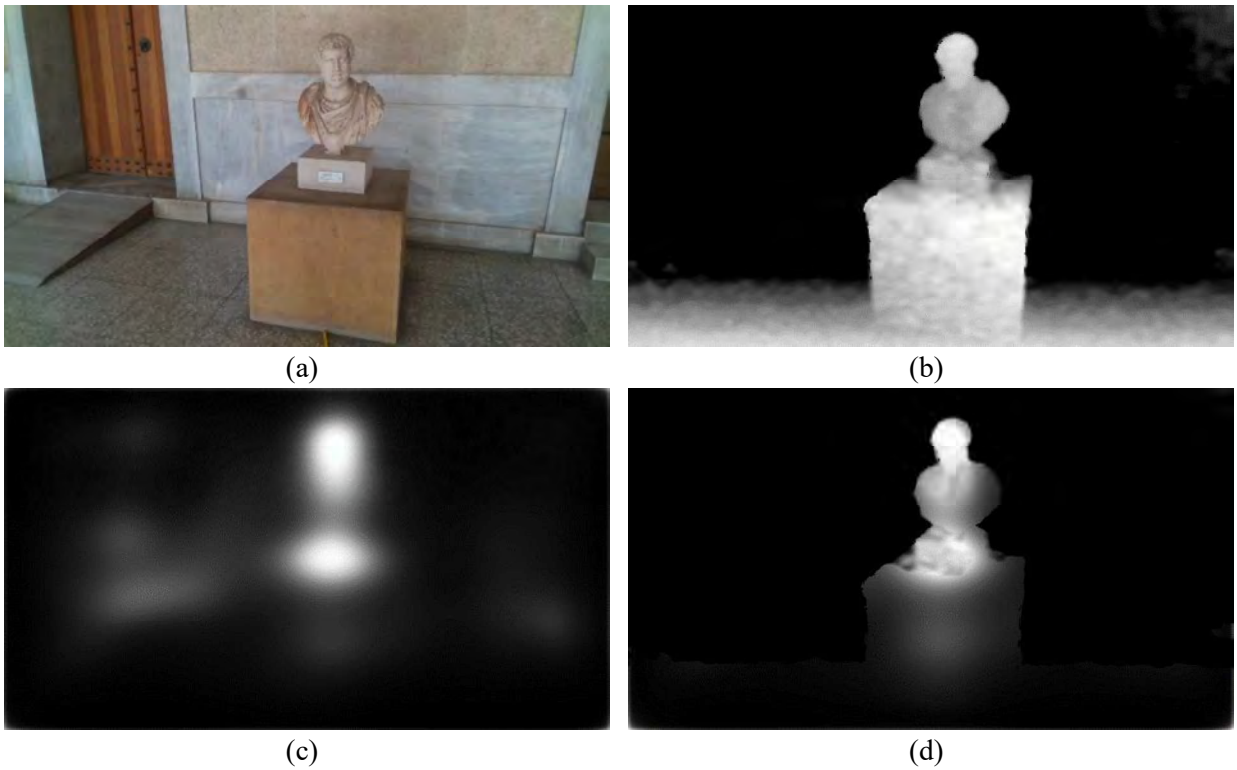


Figure 4.14. Example of the aggregation process between the saliency map SM and the high-risk map $\boldsymbol{R_M^1}$. (a) Original I_{RGB} used for the generation of the saliency map SM; (b) high-risk map $\boldsymbol{R_M^1}$ used in the aggregation; (c) saliency map S_M based on the human eye fixation on image (a); (d) the aggregation product using the fuzzy AND operator between images (b) and (c).

map R_M^i, where *i = 1, 2, 3*, using the fuzzy AND (∧) operator (Godel t-norm) (Feferman, Dawson, Kleene, Moore, & Solovay, 1998), formally expressed as:

$$F_1 \wedge F_2 \coloneqq \min\left(F_1(x,y), F_2(x,y)\right) \tag{4.5}$$

In Eq. (4.5) F_1 and F_2 denote two generic 2D fuzzy maps with values within the [0, 1] interval, and x, y are the coordinates of each value of the 2D fuzzy map. The risk maps R_M^i are, by definition, fuzzy 2D maps, since they derive from the responses of membership functions r_i on a depth map. The saliency map S_M can be considered as a fuzzy map where its values represent the degree of participation of a given pixel to the salient domain. Therefore, they can be combined with the fuzzy AND operator to produce a new fuzzy 2D map O_M^i as follows:

$$O_M^i = R_M^i \wedge S_M \tag{4.6}$$

The non-zero values of the 2D fuzzy map O_M^i (obstacle map) at each coordinate (x, y) indicate the location of an obstacle and express the degree of participation in the risk domain of the respectiveR_M^i. Figure 4.14 (d) illustrates the respective O_M^i produced using the fuzzy AND operator

with the three R_M^i. Higher pixel values of the O_M^i portray higher participation on the respective risk category and the probability of the location of an obstacle.

Theoretically, the O_M^i can be directly used to detect obstacles posing different degrees of risk to the VCP navigating in the area. However, if the orientation of the camera is towards the ground, the ground plane can be often falsely perceived as obstacle. Consequently, a refinement step is needed to optimize the obstacle detection results and reduce the occurrence of false alarm error. Therefore, a simple but effective approach for ground plane extraction is adopted. The ground plane has a distinctive gradient representation along the *Y* axis in depth maps, which can be exploited in order to remove it from the O_M^i. As a first step, the gradient of the depth map *D* is estimated by:

$$\nabla D = \left(\frac{\partial D}{\partial x}, \frac{\partial D}{\partial y}\right) \tag{4.7}$$

A visual representation of a normalized difference map $\frac{\partial D}{\partial y}$ in the [0, 255] interval can be seen in Figure 4.15. As it can be seen, the regions corresponding to the ground have smaller differences than the rest of the depth map. In the next step, a basic morphological gradient *g* (J.-F. Rivest, Soille, & Beucher, 1993) is applied on the gradient of *D* along the *y* direction $\frac{\partial D}{\partial y}$. A basic morphological gradient is basically the difference between dilation and erosion of the $\frac{\partial D}{\partial y}$ given an all-one kernel $k_{5\times5}$:

$$g\left(\frac{\partial D}{\partial y}\right) = \delta_{k_{5\times5}}\left(\frac{\partial D}{\partial y}\right) - \varepsilon_{k_{5\times5}}\left(\frac{\partial D}{\partial y}\right) \tag{4.8}$$

where δ and ε denote the operations of dilation and erosion and their subscripts indicate the used kernel. In contrast to the usual gradient of an image, the basic morphological gradient *g* corresponds to the maximum variation in an elementary neighborhood rather than a local slope. The morphological gradient is followed by consecutive operations of erosion and dilation with a kernel $k_{5\times5}$. As it can be noticed in Figure 4.15, the basic morphological filter *g* gives higher responses on non-ground regions and thus the following operations of erosion and dilation are able to eliminate the ground regions quite effectively. The product of these consecutive operations is a ground removal mask G_M, which is then multiplied with O_M^i, setting the values corresponding to the ground, to zero. This ground removal approach has been experimentally proven to be sufficient to eliminate the false identification of the ground as obstacle. A visual representation of the ground mask creation and the ground removal can be seen in Figure 4.15 and Figure 4.16, respectively.

Once the obstacle map of the depicted scene is estimated following the process described above, the next step is the spatial localization of the obstacle in linguistic values. This step is crucial for the communication of the surroundings to a VCP. For this purpose, Fuzzy Sets are utilized by the proposed methodology. Five membership functions are used to determine the location of an obstacle along the horizontal axis (*x*-axis) and 3 along the vertical axis (*y*-axis).

Initially, the boundaries of the obstacles depicted in the obstacle maps need to be determined. For the obstacle detection task, the O_M^1 obstacle map, through which the high-risk obstacles are

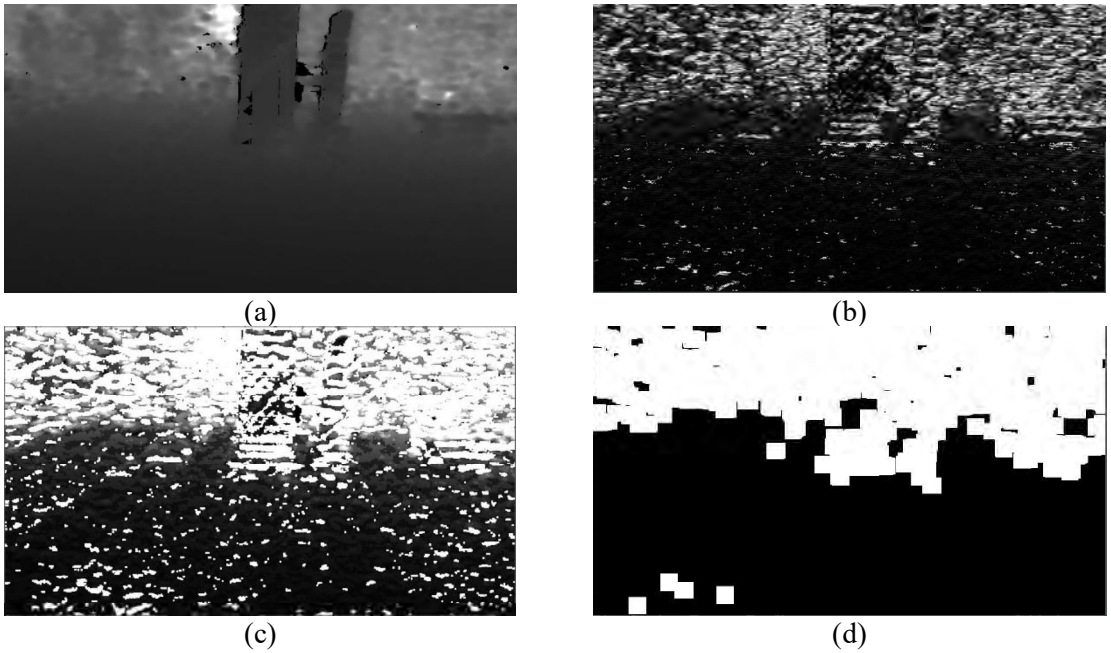


Figure 4.15. Example of the creation steps of G_M. (a) Depth map D, normalized for better visualization; (b) visual representation of the difference map Δ_M; (c) difference map Δ_M after the application of the basic morphological gradient; and (d) the final ground removal mask G_M.

represented, is chosen. Then, the boundaries b_l, where $l = 1, 2, 3\ldots$, of the obstacles are calculated using a border following the methodology presented in (Suzuki & others, 1985). Once the boundaries of each probable obstacle depicted in O_M^1 are acquired, their centers $c_l = (c_x, c_y)$, $l = 1, 2, 3\ldots$ are derived by exploiting the properties of the image moments (Kotoulas & Andreadis, 2005) of boundaries b_l. The centers c_l can be defined using the raw moments m_{00}, m_{10}, and m_{01} of b_l as follows:

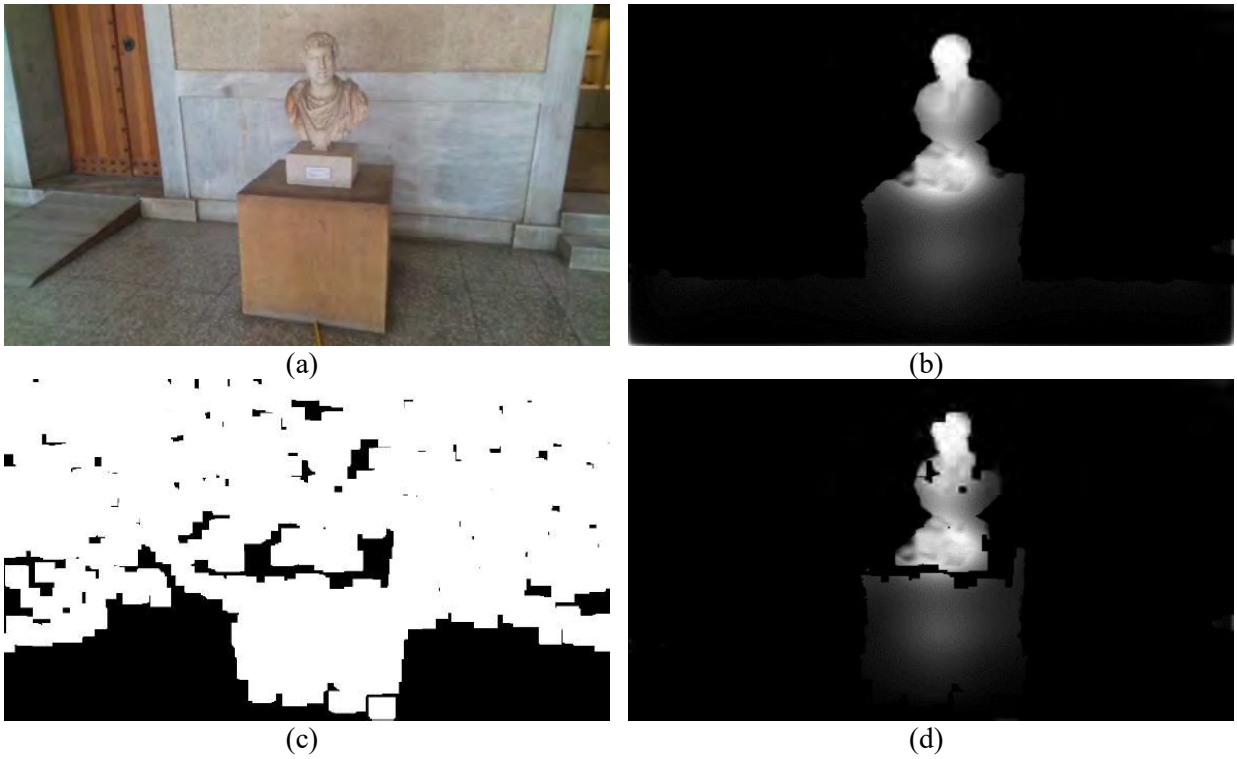
$$m_{qk} = \iint_{b_l} x^q y^k I_{RGB}(x, y) dx dy \tag{4.9}$$

$$c_l = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{4.10}$$

where $q = 0, 1, 2\ldots$, $k = 0, 1, 2\ldots$ and x, y denote image coordinates along the x-axis and y-axis respectively. An example of the obstacle boundary detection can be seen in Figure 4.17, where the boundaries of the obstacles are illustrated with green lines (Figure 4.17 (b)) and the centers of the obstacles are marked with red circles (Figure 4.17 (c)).

Once the centers have been calculated, their location can be determined and described with linguistic values using the horizontal and vertical membership functions, h_j, where $j = 1, 2, 3, 4, 5$,

and v_i, where $i = 1, 2, 3$. If the response of $h_j(c_x)$ and $v_i(c_y)$ is greater than 0.65, then the respective obstacle with a boundary center of $c_l = (c_x, c_y)$ will be described with the linguistic value that these



(a) (b) (c) (d)

Figure 4.16. Example of the ground removal procedure. In (d), the ground has been effectively removed. (a) Original I_{RGB} image; (b) corresponding obstacle map $\boldsymbol{O_M^1}$; (c) respective ground removal mask G_M; (d) masked obstacle map $\boldsymbol{O_M^1}$;

(a) (b) (c)

Figure 4.17. Example of the obstacle boundary extraction and obstacle center calculation. (a) $\boldsymbol{O_P^1}$ obstacle map used for the detection of high-risk obstacles; (b) boundary (green outline) estimation of the obstacles; (c) respective centers of the detected obstacles.

h_j and v_i represent. Additionally, the distance between object and person is estimated using the depth value of depth map D at the location of $D(c_x, c_y)$. Using this information, the VCP can be warned regarding the location and distance of the obstacle and, as an extension, be assisted to avoid it.

The obstacle map depicts probable obstacles that are salient for humans and are within a certain range. However, this can lead to false positive indications, since some obstacles, such as tree

branches, can be within a range that can be considered threatening, but at a height greater than that of the user, not affecting his/her navigation. False positive indications of this nature can be avoided using the membership function $p(h_o, h_u)$. To use this membership function, the 3D points of the scene need to be determined by exploiting the intrinsic parameters of the camera and the provided depth map.

To project 2D points on the 3D space in the metric system (meters), we need to know the corresponding depth value z for each 2D point. Based on the pinhole model, which describes the geometric properties of our camera (Heikkila & Silvén, 1997), the projection of a 3D point to the 2D image plane is described as follows:

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \frac{f}{z} \begin{pmatrix} X \\ Y \end{pmatrix} \tag{4.11}$$

Where f is the effective focal length of camera and $(X, Y, z)^{\mathrm{T}}$ is the 3D point corresponding to a 2D point on the image plane $(\tilde{u}, \tilde{v})^T$. Once the projected point $(\tilde{u}, \tilde{v})^T$ is acquired, the transition to pixel coordinates $(x, y)^{\mathrm{T}}$ is described by the following equation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} D_u s_u \tilde{u} \\ D_v \tilde{v} \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{4.12}$$

s_u denotes a scale factor, D_u, D_v are coefficients needed for the transition from the metric units to pixels and $(x_0, y_0)^{\mathrm{T}}$ is the principal point of the camera. With the combination of Eqs. (4.11)-(4.12) the projection which describes the transition from 3D space to the 2D image pixel coordinate system can be expressed as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{f D_u s_u X}{z} \\ \frac{f D_v Y}{z} \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{4.13}$$

The 3D projection of a 2D point with pixel coordinates (x, y), for which the depth value z is known, can be performed by solving Eq. (4.13) for X, Y formally expressed below (Dimitris K Iakovidis, Dimas, et al., 2018):

$$\begin{pmatrix} X \\ Y \end{pmatrix} = z \begin{pmatrix} \frac{x - x_0}{f_x} \\ \frac{y - y_0}{f_y} \end{pmatrix} \tag{4.14}$$

where $f_x = fD_u s_u$ and $fy = fD_v$. Eq. (4.14) is applied on all the 2D points of I_{RGB} with known depth values z. After the 3D points have been calculated, the Y coordinates are used to create a 2D height map H_{M} of the scene, where each value is a Y coordinate indicating the height an object at the corresponding pixel coordinate in I_{RBG}. Given the height h_u of the user, we apply the p membership

function on the height map H_M to assess the risk with respect to the height of the user. The responses of p on H_M create a 2D fuzzy map P_M as shown below:

$$P_M(x, y) = p(H_M(x, y), hu) \tag{4.15}$$

Finally, the fuzzy AND operator is used to combine O_M^i with P_M, resulting in a final personalized obstacle map O_P^i:

$$O_P^i = O_M^i \wedge P_M \tag{4.16}$$

Non-zero values of O_P^i represent the final location of a probable obstacle with respect to the height of the user and the degree of participation to the respective risk degree, *i.e* the fuzzy AND operation between O_P^1 with P_M describes the high-risk obstacles in the scenery

Experiments and Results

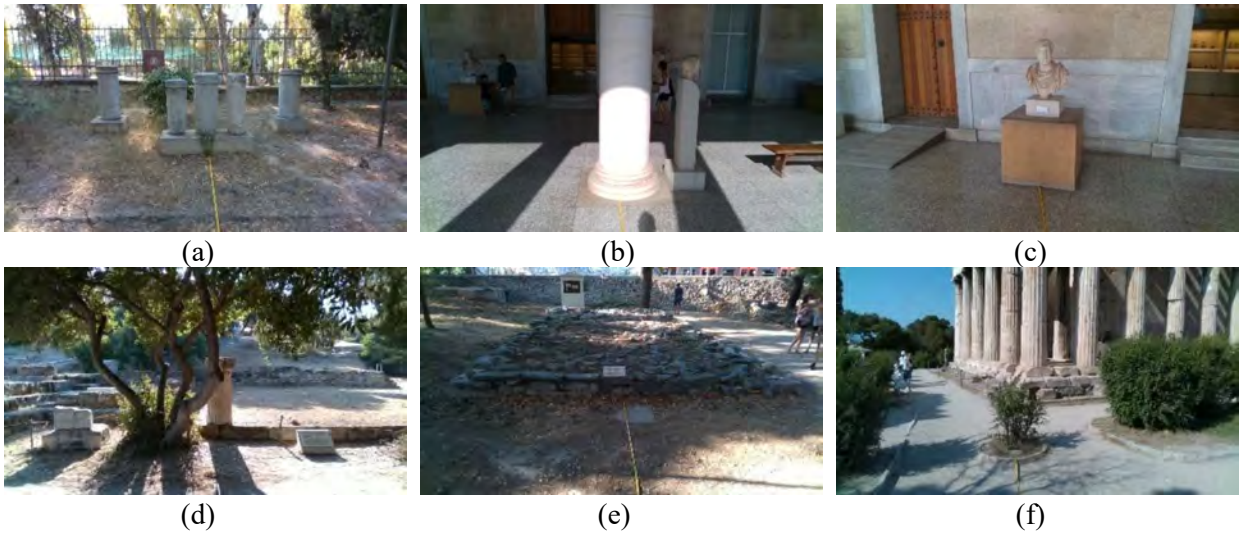


Figure 4.18. Example of the objects identified as obstacles in our dataset. (a-c) columns/artifacts; (d) tree; (e) cultural sight near the ground level; (f) small tree/bush.

To validate the proposed system, a new dataset was constructed consisting of videos captured from an area of cultural interest, namely the Ancient Agora of Athens, Greece. The videos were captured using a RealSense D435 mounted on smart glasses and were divided into two categories. The first category focused on videos of free walk around the area of Ancient Agora, and the second category on controlled trajectories towards obstacles found in the same area. The validation of the system was developed around both obstacle detection and their class recognition. When an obstacle was identified and its boundaries were determined, the area of the obstacle was cropped and propagated to the obstacle recognition network. In the rest of this section, the experimental framework will be further described along with results achieved using the proposed methodology.

The dataset composed for the purposes of this study focuses on vertical obstacles that can be found in sites of cultural interest. The dataset consisted of 15.415 video frames captured by researchers

wearing the smart glasses. In 5.138 video frames the person wearing the camera was walking towards the obstacles, but not in a range for the obstacle to be considered threatening. In the rest 10.277 video frames, the person was walking until collision, towards obstacles considered as threatening, which should be detected and recognized. The intervals determining whether an

(a) (b) (c)

Figure 4.19. Qualitative example of false ground detection as obstacle resulted using the methodology presented in (George Dimas, Ntakolia, et al., 2019) (section 4.1). In all images, the obstacles are not in a threatening distance. (a) False positive detection on dirt ground-type; (b) False positive detection on rough dirt ground-type. (c) False positive detection on tile ground-type.

Table 4.2. Confusion matrix of the methodology proposed in section 4.2

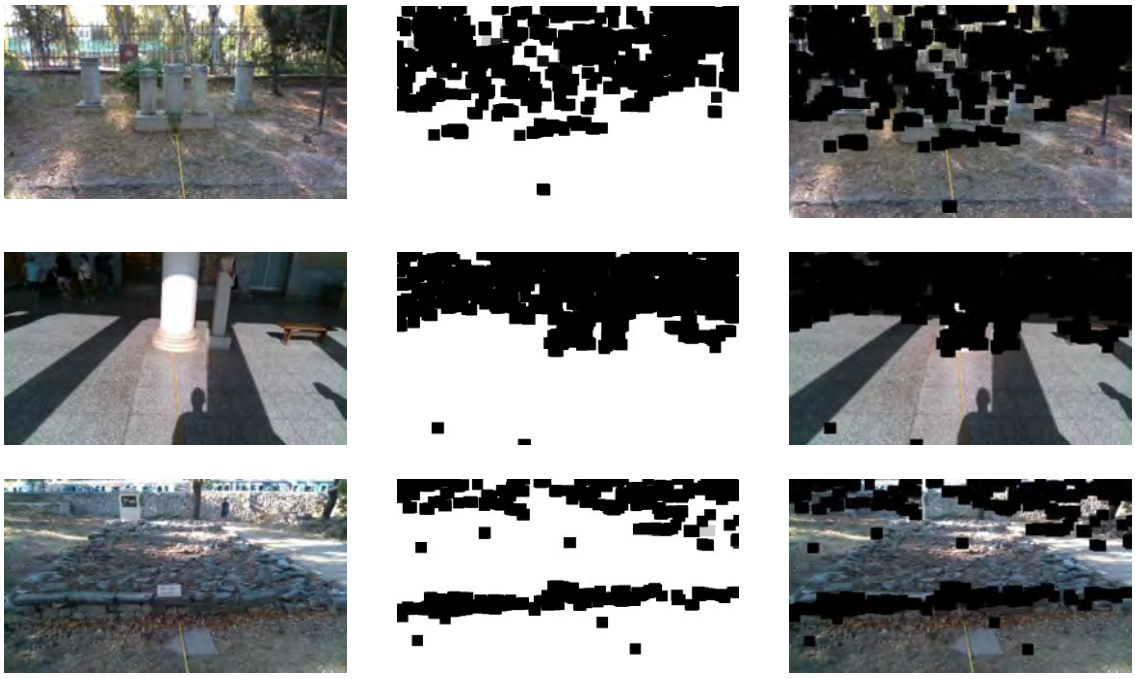
Actual	Detected	
	Positive (%)	Negative (%)
Positive (%)	55.1	9.0
Negative (%)	5.3	30.6

obstacle is considered as threatening or not were set according to the user requirements established by VCP for obstacle detection tasks in (Dimitris K Iakovidis et al., 2020). Regarding that, the desired detection distance for the early avoidance of an obstacle according to the VCP user requirements is up to 2 m.

During data collection, the camera captured RGB images, corresponding depth maps, and stereo infrared (IR) images. The D435 sensor is equipped with an IR projector which is used for the improvement of depth quality through the projection of an IR pattern that enables texture enrichment. The IR projector was used during the data acquisition for a more accurate estimation of the depth. In this study, only the RGB images and the depth maps needed for our methodology were used. The categories of obstacles visible in the dataset were columns, trees, archaeological artifacts, crowds, and stones. An example of types of obstacles included in our dataset can be seen in Figure 4.18. As previously mentioned, all data were captured in an outdoor environment, in the Ancient Agora of Athens. In addition, it is worth noting that the data collection protocol that was followed excludes any images that include human subjects that could be recognized in any way.

For the obstacle detection task, only the high-risk map was used, since it depicts objects that pose immediate threat to the VCP navigating the area. The high-risk interval of the membership function r_1 was decided to be at $0 < z < 3.5$ m. By utilizing the fuzzy sets, an immediate threat within the range of $0 < z < 1.5$ m can be identified, since the responses of r_1 in this interval are 1 and then it

degrades until the distance of 3.5 m, where it becomes 0. With this approach, the uncertainty within the interval of $1.5 < z < 3.5$ m is taken into consideration, while at the same time, the requirement regarding the detection up to 2 m is satisfied. The GAN that was used for the estimation of the saliency maps based on the human eye-fixation was trained on the SALICON dataset (M. Jiang et al., 2015).



(a) (b) (c)

Figure 4.20. Qualitative representation of the ground removal method. (a) Original I_{RGB} images; (b) Ground masks with the white areas indicating the ground plane; (c) images of (a) masked with the masks of (b).

Table 4.3. Results and quantitative comparison between the proposed and state-of-the art methodologies.

Metrics	Proposed (%)	Method (George Dimas, Ntakolia, et al., 2019) (%)	Method (Lee, Su, & Chen, 2012) (%)
Accuracy	85.7	72.6	63.7
Sensitivity	86.0	91.7	87.3
Specificity	85.2	38.6	21.6

The proposed methodology was evaluated on the dataset described above. For the evaluation of the obstacle detection methodology the sensitivity, specificity, and accuracy metrics were used. The sensitivity and specificity are formally defined as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.17}$$

$$Specificity = \frac{TN}{TN + FP} \tag{4.18}$$

where *TP* (true positive) are the true positive obstacle detections, *e.g.* the obstacles that were correctly detected, *FP* (false positive) are the falsely detected obstacles, *TN* (true negative) are frames were correctly no obstacles were detected, and *FN* (false negative) are frames that obstacles were not correctly detected.

Our method resulted in an accuracy of 85.7% on its application of the aforementioned dataset, with a sensitivity and specificity of 85.9% and 85.2%, respectively. A confusion matrix for the proposed method is presented in Table 4.2. For further evaluation, the proposed method was compared to that proposed in (George Dimas, Ntakolia, et al., 2019), which, on the same dataset, resulted in an accuracy of 72.6% with a sensitivity and specificity of 91.7% and 38.6%, respectively. The method proposed in (George Dimas, Ntakolia, et al., 2019) included neither the ground plane removal in its pipeline nor the personalization aspect. On the other hand, the proposed approach was greatly benefited from these aspects in the minimization of false alarms. As it can be seen in Figure 4.19, the dataset contains frames where the camera is oriented towards the ground, and without a ground plane removal step, false alarms are inevitable. The obstacles in Figure 4.19, were not in a range to be identified as a threat to the user; however, in Figure 4.19 (a-c), where the ground plane removal has not been applied, the ground have been falsely identified (green boxes) as obstacle. A quantitative comparison between the two methods can be seen in Table 4.3.

Qualitative results with respect to the ground detection method can be seen in Figure 4.20. As it can be observed, the methodology used for the ground plane detection is resilient to different ground types. The ground types that were found in our dataset were grounds with dirt, tiles, marble, and gravels. In addition, using such a method reduces greatly the false alarm rate when the head is oriented towards the ground plane. Even though the masking process is noisy, the obstacle inference procedure is not affected.

Current imaging, computer vision, speech, and decision-making technologies have the potentials to further evolve and be incorporated into effective assistive systems for the navigation and guidance of VCPs. The present study explored novel solutions to the identified challenges, with the aim to deliver an integrated system with enhanced usability and accessibility. Key features in the context of such a system are obstacle detection, recognition, easily interpretable feedback for the effective obstacle avoidance and a novel system architecture. Some obstacle detection methods such as (Poggi & Mattoccia, 2016) tackle the problem by incorporating deep learning methods for the obstacle detection tasks and using only the 2D traits of the images.

4.4 Self-Supervised CNN for Soft Obstacle Detection

An overview of the proposed methodology is schematically illustrated in Figure 4.21. It consists of two main components, a U-Net CNN learning model, and a supervisor, *g*, implementing the obstacle detection algorithm proposed in (George Dimas, Diamantis, et al., 2020) (section 4.2), The output of the supervisor is used to train the learning model, so that it learns to estimate the output of the obstacle detection algorithm, given an RGB input image. The learning model is

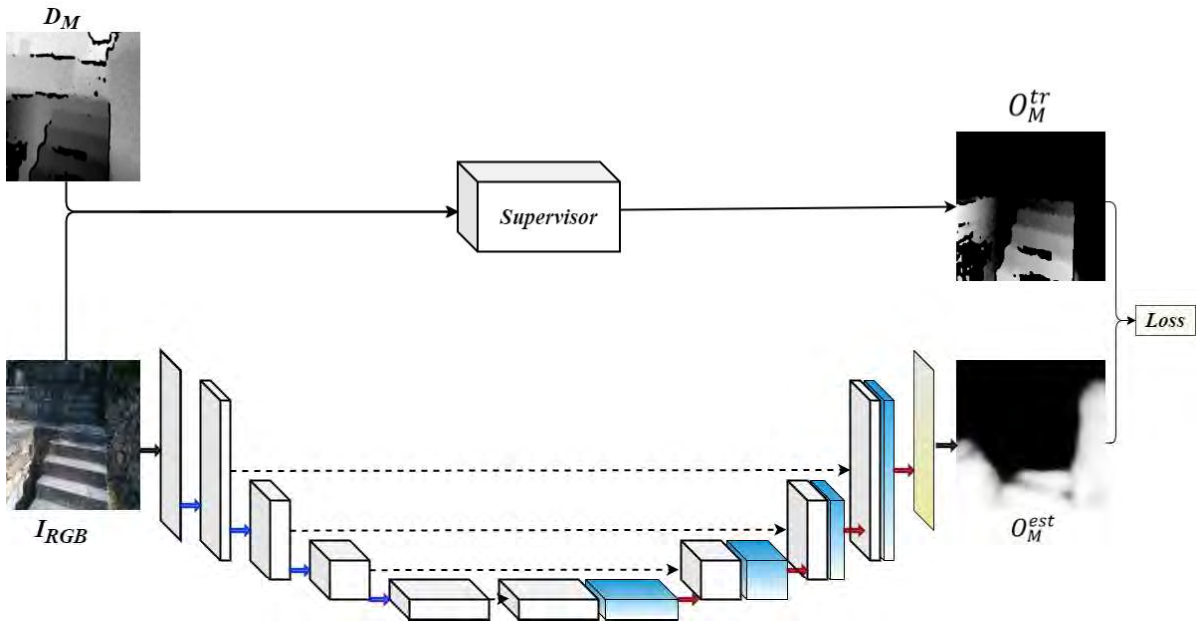
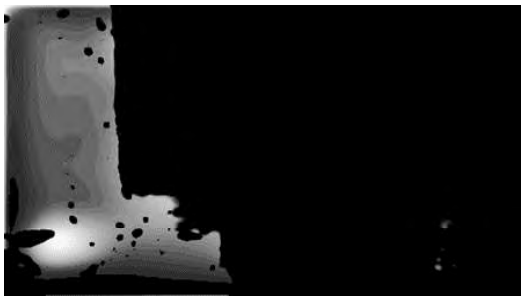


Figure 4.21. Overview of the CNN model used for self-supervised learning of the soft saliency maps.

(a)



(b)

Figure 4.22. Example of the input of the model and the corresponding training target saliency map as generated by the methodology described in section **4.2**. (a) Input RGB image; (b) training target sample.

trained in a self-supervised manner since the target training samples are automatically generated. For the training procedure, RGB-Depth image pairs are required; both the RGB and depth images are used as input to the obstacle detection algorithm to output maps illustrating regions of possible obstacles. Then the RGB image is used as input to the model which is trained to approximate the maps produced by the obstacle detection algorithm. The details of the network architecture and the self-supervised training methodology are described in the following paragraphs.

The model presented adopts an FCN-based U-Net (Ronneberger et al., 2015) architecture, comprised of an encoder part for feature extraction and a decoder part for the reconstruction of the target image. The backbone network that defines the architectural structure of the encoder is ResNet34 (He et al., 2016), pre-trained on ImageNet (J. Deng et al., 2009). The encoder is a variant of the ResNet model, composed of 34 convolutional layers with residual connections. The first convolution layer utilizes a kernel with a size of 7×7 followed by a max-pooling operation. The subsequent convolution blocks include a number of convolution layers whereas after each

block follows a max-pooling layer. The kernel size of the convolution layers is specified to be 3×3 and the selected activation function is ReLU. The structure of the decoder takes on a symmetric formation to the encoder with the max-pooling layers replaced by up-sampling layers and maintaining the skip connections of the encoder. The feature maps of the decoder are concatenated with the corresponding feature maps of the encoder after each step. The final layer of the decoder uses a sigmoid activation function that outputs the predicted image.

The U-Net model is trained in a self-supervised manner. Self-supervised training describes the process where the supervisory signal, defining the learning task of a neural network, is generated via an automatic process (Jing & Tian, 2020). Thus, target training data, are generated without human supervision or intervention. The training samples are generated automatically using the methodology described in 4.2. That approach uses fuzzy sets and a Generative Adversarial Network (GAN) to exploit both depth and RGB information for obstacle detection. The depth maps (D_M) are used for the derivation of risk maps (R_M) that their pixel values expressing high (R_M^H), medium (R_M^M) and low risk (R_M^L) regions, based on the VIP requirements assessed in (Ntakolia, Dimas, & Iakovidis, 2020). The construction of these risk maps is performed using fuzzy logic. The RGB information is used to produce saliency maps (S_M) based on human eye fixation through the GAN model. To assess regions of probable obstacle occurrence, the generated saliency map is aggregated with R_M^H through a fuzzy AND operation (Nguyen et al., 2018). This process provides a 2D obstacle map, $O_M \in [0, 1]^{M \times N}$, where its pixel values express the probability of an obstacle inclusion. Examples of this aggregation for different images are presented in Figure 4.22 (b). The U-Net model is trained to estimate the O_M given only the RGB information.

Let us consider the U-Net model as a function approximator $f(I_{RGB}) = O_M^{est}$, where I_{RGB} is a color image and O_M^{est} is an estimated obstacle map. The automatic process of the training target samples generation, as described above, can be denoted as $g(I_{RGB}, D_M) = O_M^{tr}$, where O_M^{tr} is used as a training target sample. For the training of *f*, we use a loss function incorporating the binary focal loss (L_1) (Lin et al., 2017) and the dice loss (L_2) (Sudre, Li, Vercauteren, Ourselin, & Cardoso, 2017) functions. The focal loss function introduces a modulating term to the well-known cross-entropy loss function, to focus learning on hard examples and down-weight negatives that are usually over-represented in segmentation and detection tasks. The focal loss function is formally expressed as follows:

$$L_1(p_t) = -a(1 - p_t)^{\gamma} \log(p_t) \tag{4.19}$$

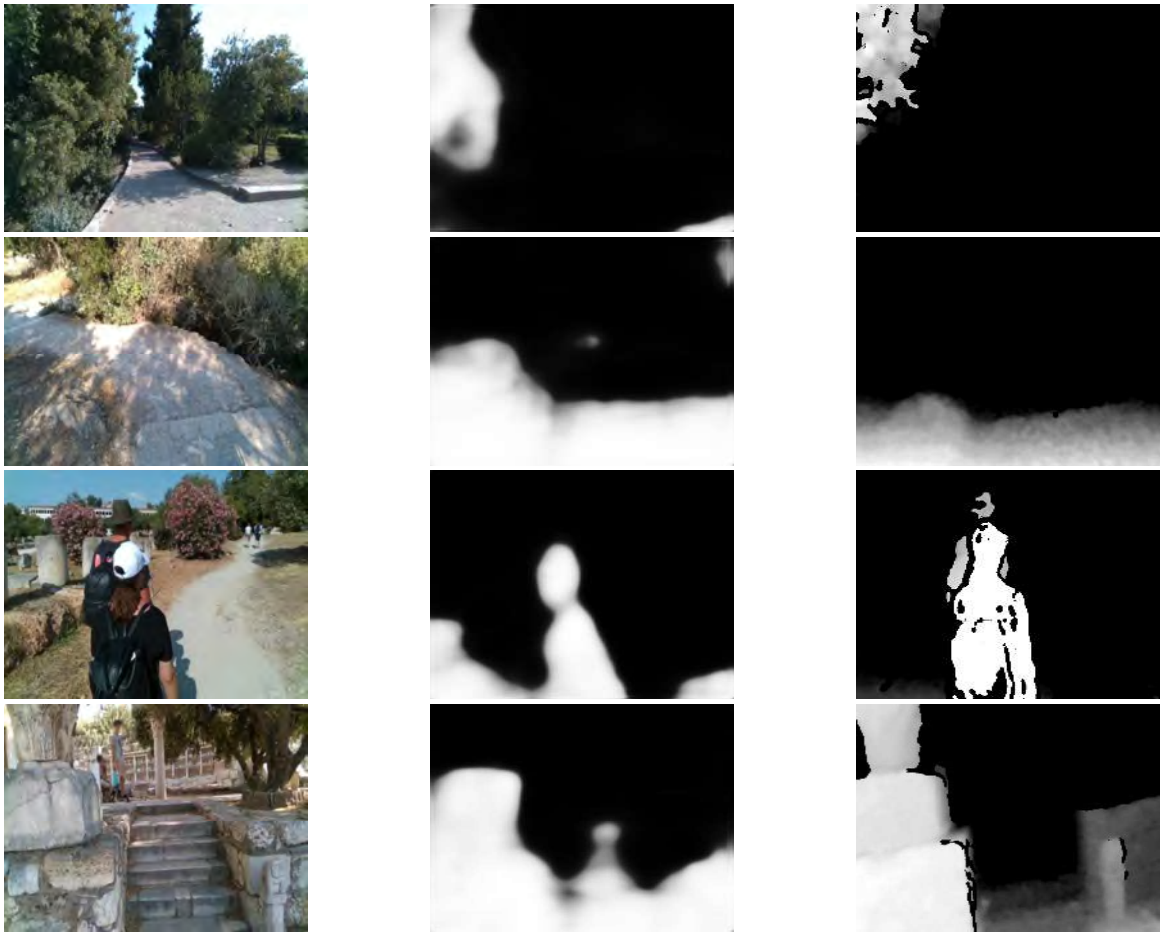
where γ is a tunable focusing parameter, a is a balanced variant, $O_M^{est}(i,j) = p$, and p_t is defined as:

$$p_t = \begin{cases} p & \text{if } O_M^{tr}(i,j) = 1 \\ 1 - p & \text{otherwise,} \end{cases} \tag{4.20}$$

Dice loss function is based on the Dice coefficient, a metric used to calculate the similarity between two images. The Dice Loss function (L_2) is formally defined as follows:

$$L_2(p, y) = 1 - \frac{2yp + 1}{y + p + 1} \tag{4.21}$$

where $O_M^{tr}(i, j) = y$, 1 is added to both the numerator and denominator for ensuring that the loss function is not undefined in edge case scenarios. So, the proposed joint loss function L is



(a) (b) (c)

Figure 4.23. Qualitative comparison between the proposed methodology and obstacle maps produced by (George Dimas, Diamantis, et al., 2020), each column represents the (a) input images; (b) Obstacle maps estimated by the proposed model ($\boldsymbol{O_M^{est}}$); (c) Obstacle maps estimated by (George Dimas, Diamantis, et al., 2020) ($\boldsymbol{O_M^{tr}}$).

constructed as the summation between L_1 and L_2. As it can be observed in Eq.(3.21) the CNN is tasked to minimize the loss L by comparing its output to the automatically generated training target sample.

$$f: argmin_f(L(f(I_{RGB}), g(I_{RGB}, D_M)) \tag{4.22}$$

The experimental evaluation was performed on three datasets consisting of RGB-D image pairs extracted from videos, captured using the Intel RealSense D435 stereo camera. The first dataset, D1, consists of images captured from the broader area of the Ancient Agora of Athens, and it

includes 861 RGB-Depth image pairs. The second dataset, D2, consists of 579 images, captured from the broader area of the Odeon of Herodes Atticus. The third dataset, D3, was used as a test set and it consists of 300 images captured from different locations from the area of Ancient Agora of Athens.

During training, the images of the respective training set were split with a proportion of 80% for training and 20% for validation. After the splitting, the images of the training set were augmented using flipping with respect to their horizontal axis. To avoid overfitting the early stopping technique was employed, monitoring the loss of the validation set.

Experiments and Results

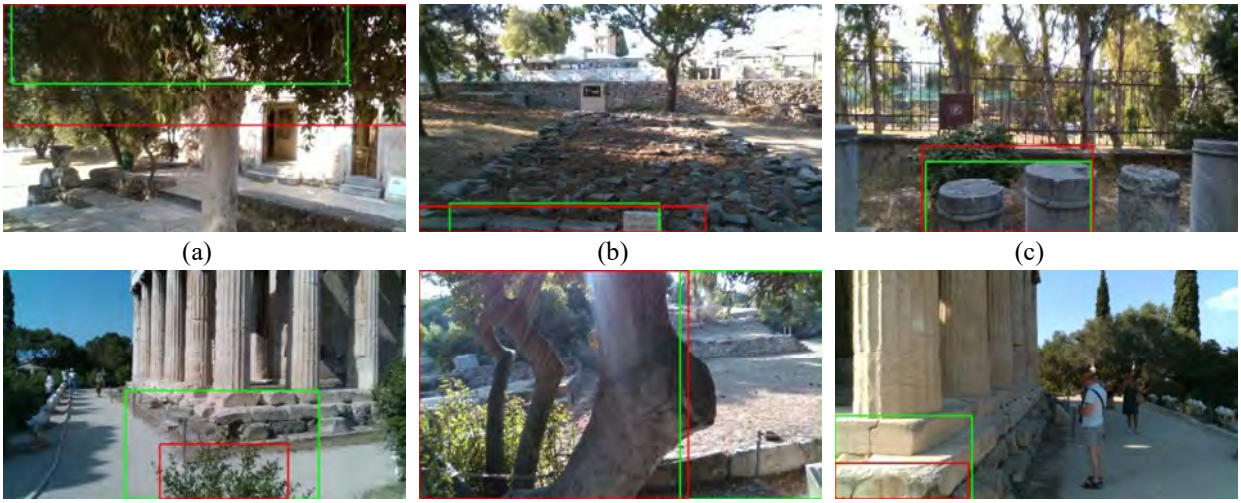


Figure 4.24. Qualitative comparison of obstacle detection performed with the proposed methodology (green rectangle) and the one proposed in section 4.2. (red rectangle). (a) tree branches. (b) ground anomaly. (c) columns. (d) small bush. (e) tree. (f) small wall.

Table 4.4. Obstacle Map Similarity in Terms of AUC-J (%)

Datasets	Input Image Size		
	96×96	*128×128*	*192×192*
D1	68.90	70.52	70.53
D2	62.01	62.07	61.01
D1,2	69.01	70.11	71.46
Average	66.64 ± 4.01	67.57 ± 4.76	67.67 ± 5.78

The proposed methodology was evaluated on the basis of two criteria: *a)* the similarity of its output with the output of the supervisor, assessed in terms of the Judd implementation of the area under receiver operating characteristic (AUC-J) (Riche et al., 2013); and *b)* the accuracy of obstacle detection in comparison to the obstacle detection approach described in 4.2. The U-Net model was

trained using three different training sets, namely D1, D2, and a joint dataset composed of both D1 and D2 datasets (D1,2). All model instances were evaluated on the same test dataset D3, which does not have any overlap with D1 and D2. This procedure was followed to investigate the generalization capacity of the proposed methodology when trained and employed on images deriving from different locations.

Table 4.5. Obstacle Detection Accuracy (%)

Datasets	**Input Image Size**		
	96×96	*128×128*	*192×192*
D1	70.92	78.37	72.06
D2	54.59	68.79	66.31
D1,2	61.44	71.07	66.69
Average	62.32 ± 8.20	72.74 ± 5.01	68.35 ± 3.21

Table 4.6. Time Performance Comparison (image/ms)

Methodology	**Input Image Size**		
Self-Supervised	*96×96*	*128×128*	*192×192*
	43.88 ± 3.01	43.87 ± 3.10	45.61 ± 6.47
Supervisor (section **4.2**)	390.04 ± 20.08		

To investigate the performance with respect to the input image size, we trained and evaluated the U-Net model with images at different scales, namely 96×96, 128×128 and 192×192. Table 4.4 summarizes the performance of all model instances, in terms of AUC-J. As it can be observed, the best model performance was achieved with 128×128 input size and training set D1. Notably, the performance variation with respect to the size of the input image is negligible. A qualitative comparison between the obstacle maps that were estimated by the proposed methodology and the methodology of 4.2 is illustrated in Figure 4.23. It can be noticed that the U-Net model can successfully approximate the respective obstacle maps.

The obstacle detection accuracy of the proposed methodology was also evaluated on dataset D3, with all model instances, *i.e.,* the models trained on D1, D2 and D1,2, for different input image sizes. Table 4.5 summarizes the results obtained, in comparison to the methodology of 4.2, which was 85.7% on the same dataset. It can be noticed that the best performance was achieved when the model was trained on D1 dataset and with an input image size of 128×128. A qualitative assessment of the detection results is presented in Figure 4.24.

In addition, we have performed a comparison between the two methodologies with respect to their time-performance. Both methods were evaluated on the same computer equipped with an AMD Ryzen 5 3400G 3.70GHz processor, 16.00 GB RAM and the NVIDIA 2060 Super GPU. The results are summarized in Table 4.6. The minimum execution time of the proposed methodology

was 43.87 ± 3.01 ms/image, whereas the execution time of the methodology presented in 4.2 was 390.04 ± 20.08 ms/image on average. These results indicate that by using a single CNN model for obstacle detection, a navigational system can be benefited considerably in terms of time performance. It is worth noting that the differences between the execution times using the different image sizes tested, is negligible.

4.4. Discussion and Future Work

In conclusion, section 4 introduces various novel methods for obstacle detection where the 3D information acquired using an RGB-D sensor has been used for assessing the risk imposed to a user in the scenery by processing the depth values of the scene with the employment of fuzzy sets. The human eye fixation was also taken into consideration, estimated by a GAN, in terms of saliency maps. The fuzzy aggregation of the risk estimates and the human eye fixation resulted to the efficient detection of obstacles in the scenery.

In detail a preliminary and extended approach to computationally assistive navigation in terms of obstacle detection and avoidance have been described in sections 4.1 and 4.2, respectively. In contrast to other depth-aware methods, such as this proposed by (Lee et al., 2012), the obstacles detected with our approach are described with linguistic values regarding their opposing risk and spatial location, making them easily interpretable by the VCP. In addition, the proposed methods do not only extract obstacles that are an immediate threat to the VCP, *e.g.,* these with non-zero responses from the high-risk membership function r_1, but also obstacles that are of medium and low risk. Therefore, all obstacles are known at any time, even if they are not of immediate high risk.

The personalization aspects of the methodology presented in section 4.2, alongside with the ground plane detection and removal, provides a significant lower false alarm rate when compared to its preliminary counterpart (section 4.1). Furthermore, this method can detect and notify the user about partially visible obstacles with the condition that the part of the obstacle is: *a)* salient, *b)* within a distance that it would be considered of high risk and *c)* in a height that would be affecting the user. The overall accuracy of the proposed method was estimated to be 85.7%, when the methodology proposed in section 4.1 resulted to an accuracy of 72.6%, based on the dataset described in the experiments and results of section 4.2. Additionally, in contrast to other methodologies such as (Suresh et al., 2017)(Cheng, Wang, Bai, & Xu, 2019; Kaur & Bhattacharya, 2018)(Bai et al., 2019; Neela Maadhuree, Mathews, & Rene Robin, 2020), the proposed obstacle detection system is solely based on visual cues obtained using only an RGB-D sensor, minimizing the computational and energy resources required for the integration, fusion, and synchronization of multiple sensors.

Additionally, a novel self-supervised CNN-based system was proposed, trying to simplify the obstacle detection methodology of 4.2, capable of efficiently detecting regions of possible high-risk obstacles given an RGB image as input, contrary to other methodologies that are based on

multiple sensors. The proposed system has a significantly lower execution time per image when compared to an RGB-D-based obstacle detection algorithm; thus, it can contribute to more efficient, less complex navigational systems for VIPs. The results obtained by this study lead to the following conclusions: a) the proposed methodology approximates the obstacle detection performance of its supervisor algorithm presented in section 4.2, while achieving significantly lower execution times; b) the obstacle detection performance of the proposed methodology is not significantly affected by the input image size used. Therefore, the computational complexity can be further reduced by using smaller input images; c) self-supervised learning can be effectively used for developing lower-complexity saliency estimation and classification systems. The methodology presented can be considered as a first step, towards the development of more efficient assistive navigation systems.

Future work includes further improvement of the obstacle detection performance of our self-supervised methodology to better approximate its supervisor. Also, the perspectives for adaptation of the proposed methodology for other application domains are promising.

Chapter 5
Visual Measurements

Visual measurements (VMs) are a research topic that has received a lot of attention in recent years. VMs employ 3D cues extracted from images to perform measurements in physical units. Such measurements may include size, depth, travel distance, and so on, and they frequently necessitate the use of stereoscopic cameras or additional sensors, such as laser-based sensors. (Y. Li & Wang, 2020). The majority of the current VM methods that do not rely on such sensors, are based on information extracted from consecutively captured images (Scaramuzza & Fraundorfer, 2011; Zheng, Tang, & Liu, 2018). VMs can be used in a wide range of domains, including autonomous navigation (Aqel, Marhaban, Saripan, & Ismail, 2016; Balamurugan, Valarmathi, & Naidu, 2016; George Dimas, Diamantis, et al., 2020) and biomedicine (George Dimas, Bianchi, et al., 2020; Dimitris K Iakovidis, Dimas, et al., 2018; F. Mahmood, Chen, Sudarsky, Yu, & Durr, 2018). The difficulty in accurately estimating the size and depth of an object from visual data stems from the projective nature of image formation, which is bounded by the geometric properties of the camera (Hartley & Zisserman, 2003). Many depth prediction approaches, such as Structure from Motion (SfM) and deep learning (DL), have been suggested in the literature. SfM approaches can forecast depth at a limited scale. DL-based techniques rely on big, annotated training datasets. In addition, they typically require prior knowledge of an adequate scaling factor in order to convert their estimates into physical units. (Godard et al., 2019; G. Huang et al., 2017). As a result, these limitations impede their performance in unconstrained conditions (*in-the-wild*) (Alhashim & Wonka, 2018).

An important method coping with single image size measurements, called Single View Metrology (SVM), has been proposed by Criminisi *et al.* (Criminisi, Reid, & Zisserman, 2000). The geometric properties of the 3D-2D projection are used in that method to compute the height of an object given the height of the camera and the horizon line. Researchers have been inspired by SVM's core principle to develop a unified solution for single image size measurements. (Hoiem, Efros, &

Table 5.1. State-of-the-art methodologies for Visual Measurements

Methods	Characteristics				
	Size Measurements	*Depth Prediction*	*Single Image*	*Uncertainty Aware*	*Deep Learning*
Criminisi et al., 2000	✓	-	✓	-	-
Hoiem et al., 2008	✓	-	✓	-	-
Liu et al., 2010	-	✓	✓	-	-
Chen et al., 2016	-	✓	✓	-	-
Alhashim et al., 2018	-	✓	✓	-	✓
Jiao et al., 2018	-	✓	✓	-	✓
Godard et al., 2019	-	✓	✓	-	✓
Zhu et al., 2020	✓	-	✓	-	✓
Proposed	✓	✓	✓	✓	-

Hebert, 2008; Zhu et al., 2020). However, SVM requires that the horizon line is known *a priori* (and this can be challenging, *e.g.*, indoors).

The research that is presented in this section is motivated by the fact that single image VMs can be extremely valuable in the context of various applications, such as the navigation of autonomous vehicles, or the assessment of findings in medical images. Furthermore, they can contribute to the creation of lower-cost, more adaptable solutions with less hardware complexity. Henceforth, two novel methodologies are proposed, one designed for general applications *in-the-wild* and one designed for biomedical applications which is benchmarked in the context of endoscopy in the gastrointestinal (GI) tract.

The first methodology, named Virtual Grid Mapping (VGM), uses the inverse projection mapping as its foundation for visual size measurements based on a single image. Instead of attempting to deduce 3D information from 2D image space, an automatic grid of virtual 3D points is generated and projected to the 2D image plane, allowing the establishment of initial direct correspondences between 3D and 2D points. With these correspondences, the size and distance of an object of

interest from the camera can be inferred. Unlike previous methods, the proposed one accounts for camera calibration and positioning uncertainty by employing an adaptive fusion process to improve the robustness of the initial depth estimation, that is subsequently used in size estimation. In comparison to SVM, the proposed VGM method introduces an alternative, simpler, and more reliable approach that does not require knowledge of the horizon line in the input images, nor any complex training process on large datasets, as in the case of DL methods.

The main contributions of VGM can be summarized as follows: a) a novel, straightforward method for measuring object sizes (and depth) in natural images based on a virtual grid; b) an adaptive fusion methodology that deals with the uncertainty caused by camera calibration and positioning; c) The evaluation and comparison of the proposed method with other state-of-the art approaches in simulated, controlled, and *in-the-wild* conditions.

This section also introduces a methodology similar to VGM developed for biomedical applications aiming to *in-vivo* measurement of the size of an endoscopically observed object of interest in the GI tract. Beyond the state-of-the-art, the proposed methodology is based on a single image, rather than multiple consecutively captured images of the same object as in previous approaches (Dimitris K Iakovidis et al., 2019). Given the intrinsic parameters of the camera it enables the measurement of linear segments (*e.g.*, along the length or width of the object) in physical units, *e.g.*, in mm. The evaluation of the proposed approach was performed with a three-dimensional 3D-printed colon phantom, in which various objects of known size were placed in order to be measured.

5.1 Virtual Grid Mapping for Single-Image Measurements

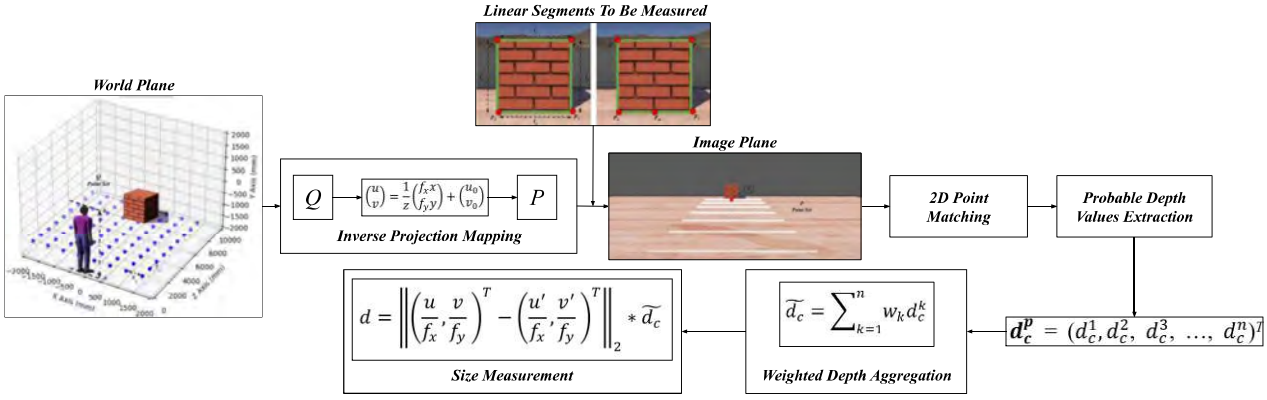


Figure 5.1. Overview of the pipeline of the proposed methodology

The proposed methodology can be divided in two parts. The first part involves calculating the distance between an object of interest and a camera (depth). This is accomplished by establishing 3D-2D direct correspondences by projecting to the image plane a grid of automatically generated virtual 3D points, Q. The feasibility of this step relies on the availability of the camera model and intrinsic parameters. With these initial 3D-2D correspondences and a defined linear segment to be measured within an image, depth information can be extracted. With the depth information known, we can proceed to measure the linear segment. Another assumption that is considered for size measurements using the proposed method, as with other single image approaches, is that the object of interest is lying on the ground (Criminisi et al., 2000; Hoiem et al., 2008; Zhu et al., 2020). The second part of the proposed method assumes that the initial depth estimation is only a rough approximation of the true depth due to uncertainty caused by the accuracy of camera calibration and the camera positioning. To deal with this, a set of possible depth values is considered, and they are fused together using different weights that are adapted through an optimization process. Figure 5.1 depicts an overview of the proposed methodology.

Virtual Grid Mapping: Let us consider an object in the 2D image plane, contained in a rectangular bounding box, b. The bounding box has four linear segments $b = (l_1, l_2, l_3, l_4)^T$, which are defined by a set of points p_j , $j = 1, 2, 3, 4$, on the image plane, *e.g.,* $l_1 = (p_4 , p_1)^T$, each of which have a pair of point coordinates $(u, v)^T$ in the image. An example bounding box is illustrated in Figure 5.1. Overview of the pipeline of the proposed methodology Figure 5.1. Our objective is to calculate the size d of a linear segment defined by two points p and p' in world units. To achieve this, the distance d_c between the camera and the object of interest is required.

The point of origin in the world coordinate system is defined by the camera, positioned at a height h ($x_c = 0$, $y_c = h$, $z_c = 0$); hence, points that lay on the ground plane are positioned on $y_g = -h$. For the purpose of this study the pinhole camera model is adopted as described in (Heikkila & Silvén,

1997). Thus, the projection of a 3D world point to the 2D image plane, can be formally expressed as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{f D_u s_u x}{z} \\ \frac{f D_v y}{z} \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \qquad (5.1)$$

where f is the effective focal length of the camera, s_u is a scale factor, D_u and D_v are coefficients enabling the transition from the metric unit system to pixels, and $(u_0, v_0)^T$ denotes the principal point of the camera. It should be noted that the camera lens is assumed not to introduce any distortion effect to the images. By denoting $f_x = fD_u s_u$ and $f_y = fD_v$ Eq. (5.1) can be rewritten in a more intuitive form as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{z} \begin{pmatrix} f_x x \\ f_y y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \qquad (5.2)$$

Since the projection from the 3D world to the 2D image plane is defined, it is now possible to project virtual 3D points q to the 2D image plane using Eq. (5.2). Let Q be a set of generated virtual 3D points $q = (x, y_g, z)^T$, where $x, z \in \mathbb{R}$ are coordinates in the *XYZ* world coordinate system (Figure 5.1). The *X*- and *Y*-axes are along the width and height of the 3D scene while the *Z*-axis is along its depth. In addition, the generation of points $q \in Q$ is constrained as follows:

$$mod(x, s_x) = 0, -a \leq x \leq a \qquad (5.3)$$

$$y_g = -h \qquad (5.4)$$

$$mod(z, s_z) = 0, 1 \leq z \leq b \qquad (5.5)$$

where $a, b \in \mathbb{R}$ are variables that set the maximum and minimum values of x and z coordinates, *i.e.*, they delimit the spatial span of the points along the *X*- and *Z*-axis. The variables $s_x, s_y \in \mathbb{R}$ denote the density of the 3D points along the *X*- and *Z*-axis, respectively, whereas *mod* denotes the modulus operation. Hence, the set Q can be considered as a 3D point cloud, where each point $q \in Q$ can be projected to the 2D image plane by applying Eq. (5.2) $\forall q \in Q$. The projection of points $q \in Q$ to the 2D image plane leads to the construction of a 2D point set P that defines the ground plane in the image space. In this way, each point $p \in P$ corresponds to a known 3D point. A graphical representation of point sets Q and P is illustrated in Figure 5.1.

Depth Estimation: As mentioned in the previous section, the object to be measured, is contained within a rectangular bounding box, b, where the l_4 is the linear segment of its therefore, all the points of l_4 lie on the ground plane. Initially, to be able to measure the object, we need to find the distance d_c between the object and the camera. To achieve this the middle point of l_4 is estimated as $p_m = \frac{1}{2}|p_4 + p_3|$. From Eq. (5.2) it can be derived that 3D points with the same coordinates x, y and z, will have the same coordinates on the image plane. Hence, to find the approximate depth

value d_c the Euclidean distance between p_m and all the points of P is calculated. Then the point in P that minimizes that distance is selected:

$$p_c = \min\left(\|p_i - p_m\|_2\right) \tag{5.6}$$

where $i = 1, 2, 3, \ldots$ is the index of each point in set P. From Eq. (5.6) it can be easily derived that $p_m \sim p_c \in P$ and thus, p_m has approximately a corresponding virtual point $q_m \in Q$. The point q_m is the 3D projection of p_c , which according to the point projection model, its z coordinate is a close approximation of the z coordinate of p_m, *i.e.*, $z = d_c$. However, the density of the virtual grid and the positioning of the camera introduce a degree of uncertainty to the initial estimation of d_c. To cope with that, a refinement step of d_c is proposed. Instead of using directly the initial distance d_c from the point p_c determined by Eq.(6.6), a set of n different distances d_c^k, $k = 1, 2, 3, \ldots n$, are selected from a respective set of n points p_c^k, $k = 1, 2, 3, \ldots n$, closer to p_m. Thus, a vector $d_c = (d_c^1, d_c^2, d_c^3, \ldots, d_c^n)^{\mathrm{T}}$ is formed, and all the distances d_c^k are fused together using a weight vector $w = (w_1, w_2, w_3, \ldots, w_n)^{\mathrm{T}}$, $w_n \in \mathbb{R}$, as follows:

$$p_c = \min\left(\|p_i - p_m\|_2\right) \tag{5.7}$$

Weights Estimation: Changes of the position of the camera and the accuracy of the calibration affects the projection of the virtual grid. As a result, the initial estimation of d_c can be based on wrongly selected points of the virtual grid. Hence, a refinement step is required in order to estimate a more precise distance $\widetilde{d_c}$ between the camera and the object of interest.

The estimation of $\widetilde{d_c}$ requires the approximation of a weight vector w, as it was mentioned in the previous section. The purpose of these weights is their application on a depth fusion procedure (Eq. (7)) where probable depth values are aggregated to estimate a more accurate estimate of the final distance between the camera and the object to be measured. Hence, a loss function L needs to be minimized with respect to an objective for the approximation of w.

The training objective that aims to approximate w requires the correct estimation of a set of ground truth distance d_{gt}, between the camera and the object. This is achieved by using a set of probable depth values in d_c along with a set of trainable weights w for each ground truth distance d_{gt}. The function f_d that we want to approximate is defined as follows:

$$p_c = \min\left(\|p_i - p_m\|_2\right) \tag{5.8}$$

Where x is a $1 \times n$ vector of depth values and $w = (w_1, w_2, w_3, \ldots, w_n)^T$. Given Eq. (5.8) the loss function L that we want to minimize with respect to w is:

$$L(d_{gt}, f_d(d_c\ ;\ w)) = \left\|d_{gt} - f_d\left(d_c\ ; w\right)\right\|_1 \tag{5.9}$$

For the approximation of w using Eq. (4.10), the Nelder-Mead optimization algorithm was used (F. Gao & Han, 2012).

Size Measurements: The estimation of $\widetilde{d_c}$ that denotes an uncertainty-aware estimator of the distance between the camera and the object of interest. By solving Eq. (5.2) with respect to *x* and *y* using as *z* the estimated $\widetilde{d_c}$, the following equation can be derived:

$$\binom{x}{y} = \widetilde{d_c}\begin{pmatrix}\frac{u-u_0}{f_x}\\ \frac{v-v_0}{f_y}\end{pmatrix} \tag{5.10}$$

By applying Eq. (5.10) to p and p' their 3D counterparts $q = (x, y, \widetilde{d_c})$, $q' = (x', y', \widetilde{d_c})$ can be calculated. Then, d in the 3D space is computed by the Euclidean distance $d = \|q - q'\|_2$, which can be expressed with respect to p and p', as follows:

$$d = \frac{\|p - p'\|_2 * \widetilde{d_c}}{f_{pixels}} \tag{5.11}$$

In Eq. (5.11) f_{pixels} is equal to f_x and f_y when the linear segment defined by p and p' is parallel to the *X*- and *Y*-axis, respectively. In the case where the linear segment is not parallel to either axis, Eq. (5.11) takes the following general form:

$$d = \left\| \left(\frac{u}{f_x}, \frac{v}{f_y}\right)^T - \left(\frac{u'}{f_x}, \frac{v'}{f_y}\right)^T \right\|_2 * \widetilde{d_c} \tag{5.12}$$

Experiments and Results

For the evaluation of the proposed methodology and its quantitative comparison against other single-image based approaches three datasets were used; a) a dataset constructed in a simulated environment, using the Webots robot simulation platform (Michel, 2004; Webots, n.d.); b) a dataset consisting of natural images, captured using a stereoscopic camera (Intel RealSense® D435) of various objects with known dimensions and in known distances from the camera and c) images captured *in-the-wild* illustrating monuments of known dimensions.

The evaluation process of the proposed methodology was performed through size measurements with respect to the height and width of the objects. The evaluation metric that was used, was the Mean Absolute Percentage Error (MAPE) defined as:

$$MAPE = 100\% \left(\frac{1}{N} \sum_{i=1}^{N} \frac{|gt_i - est_i|}{gt_i} \right) \tag{5.14}$$

where N denotes the total number of samples, gt_i and est_i are the ground truth and estimated value of the i^{th} sample, respectively defined as:

$$gt_i = height_i^{gt} + width_i^{gt} \tag{5.15}$$

$$est_i = height_i^{est} + width_i^{est} \quad (5.16)$$

By using MAPE we can directly compare the performance of our measurement methodology among objects of different sizes. Following the methodology described in this section, we initially estimated the weights that have been used for the depth estimation procedure. To estimate the weights that were used to perform measurements in the simulated environment, we used images of a 3m cubic object captured from a view angle of 0°. These images were not included in the evaluation of the proposed methodology and were only exploited for the approximation of weights. The same weights were used for the measurement procedure using the natural image dataset and they were slightly adjusted for the case of the *in-the-wild* measurements.

Two different experiments were performed in the simulated environment. In the first experiment, a series of measurements was conducted, utilizing a controlled experimental setup including a dataset that was constructed in the simulated environment, consisting of 5 cubic objects with sizes of 0.3, 0.5, 1.0, 2.0 and 3.0 meters. These objects were captured from different view angles and different distances. In detail, for each object, 30 images were captured from various distances in the interval of [1, 10] meters. For each distance, each object was placed in 4 different view angles with respect to the camera, *i.e.,* 0°, 5°, 15° and 20°, resulting to a total of 600 images.

In the second experiment, the proposed methodology was evaluated through a Montel Carlo experimental setup where the size and position of the cubic objects has been randomly sampled for a total of 10,000 iterations. The sampling intervals regarding the position and size have been chosen so that the entire object could be visible within the frame of the image that was captured by the simulated camera object.

The simulated camera object that was used in both experimental setups, was parameterized to approximate the Intel RealSense® D435 RGB sensor. The intrinsic parameters of the simulated camera were estimated by the simulation platform throughout the simulation. The camera was mounted on a human model, in the height of the face since most images are being captured while holding the camera in this way (Hoiem et al., 2008).

The human model had a height of 1.75m and the camera was placed with zero tilt, and its *X*-*Z* plane defined by its *X*- and *Z*-axis be parallel to the ground plane. The bounding boxes of the objects to be measured, were extracted using the object recognition capabilities of Webots camera object. Using the object recognition module of Webots it was possible to know at any moment the relative distance of the object with respect to the camera, its position on the 3D environment and its position and dimensions on the image plane. An illustration of the simulated experimental setup is illustrated in Figure 5.2.

The overall MAPE achieved on the controlled simulated dataset (SD), that was scored by the proposed methodology, considering all the objects measured from all the angles was 3.41% ± 1.30% (Table 5.2). Additionally, we have evaluated our method with respect to the impact of the weighted depth approximation by performing measurements without the incorporation of the

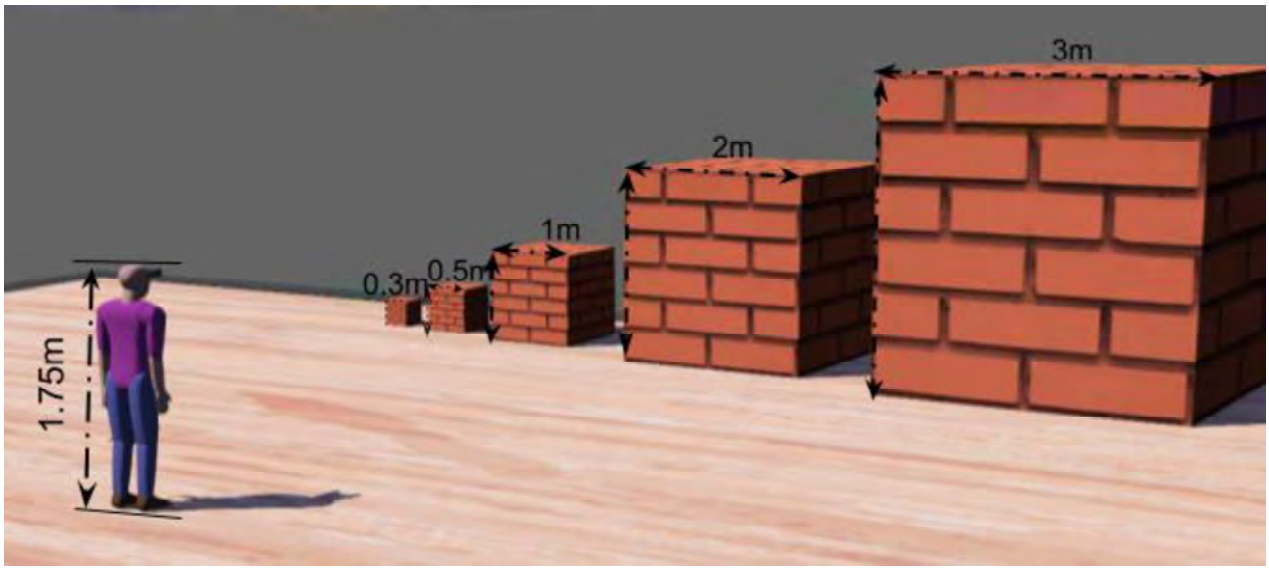


Figure 5.2. Illustration of the virtual environment along with the simulated objects that were measured

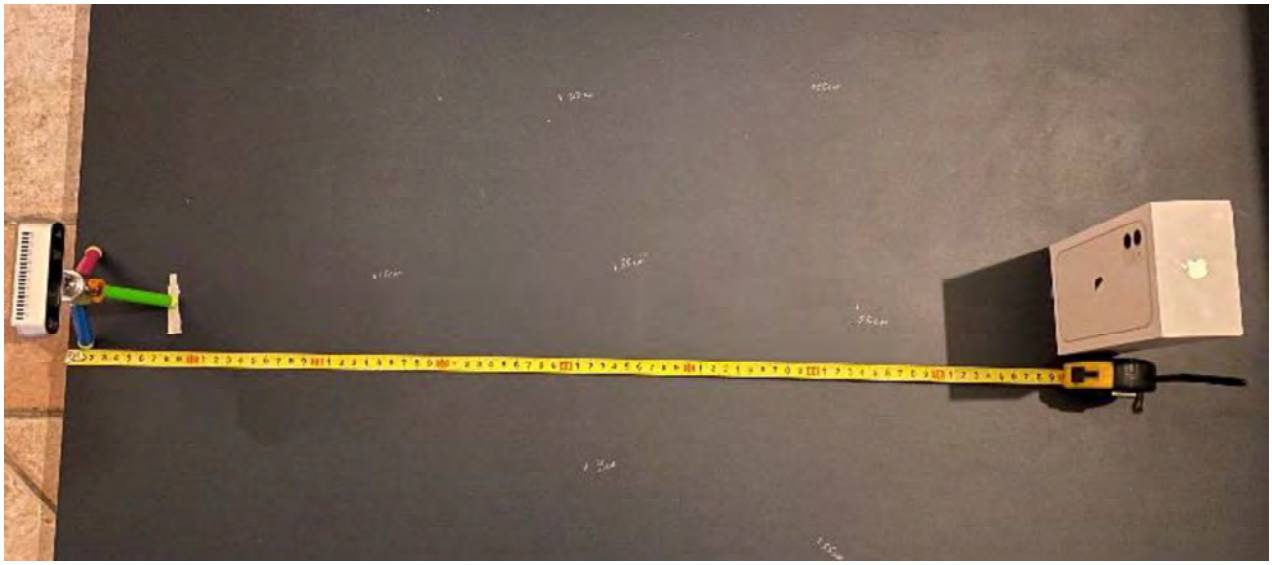


Figure 5.3. Visual representation of the experimental setup from which the natural image dataset derived

weighted aggregation step. The non-weighted approach achieved a MAPE of 4.62% ± 2.49% proving that the weighted aggregation step significantly improves the performance of the proposed method (Table 5.2). A detailed performance analysis is presented in Table 5.3 and Table 5.4 where the performance with respect to the object size (Table 5.3) and view angle (Table 5.4) is presented.

Regarding the Monte Carlo experimental setup, the proposed methodology scored a MAPE of 5.21% ± 5.32% whereas the non-weighted version achieved a MAPE of 6.15% ± 5.98. Once again, the weighted approach seems to be more consistent achieving a lower MAPE in a large-scale evaluation.

Table 5.2. Average MAPE on SD

Methods	Metrics	
	MAPE (%)	*Std. (%)*
Weighted	3.41	1.30
Non-Weighted	4.62	2.49
SVM (Criminisi et al., 2000)	4.29	1.96

Table 5.3. Average MAPE per object of SD

Object Size (*m*)	Methods					
	Weighted		*Non-Weighted*		*SVM* (Criminisi et al., 2000)	
	MAPE	Std.	MAPE	Std.	MAPE	Std.
0.3	3.21	0.44	3.83	0.49	4.17	0.53
0.5	4.15	0.53	5.40	0.59	5.70	0.55
1	5.33	0.60	6.71	0.64	6.98	0.85
2	1.95	0.15	4.62	4.99	2.09	0.19
3	2.42	0.19	2.52	0.12	2.52	0.19

Table 5.4. Average MAPE per view angle on SD

View Angle	Methods					
	Weighted		*Non-Weighted*		*SVM* (Criminisi et al., 2000)	
	MAPE	Std.	MAPE	Std.	MAPE	Std.
0°	3.05	1.07	3.69	1.68	3.78	1.69
5°	3.53	1.53	5.90	3.79	4.11	2.04
15°	3.45	1.47	4.39	2.13	4.47	2.21
20°	3.62	1.47	4.50	2.12	4.80	2.36

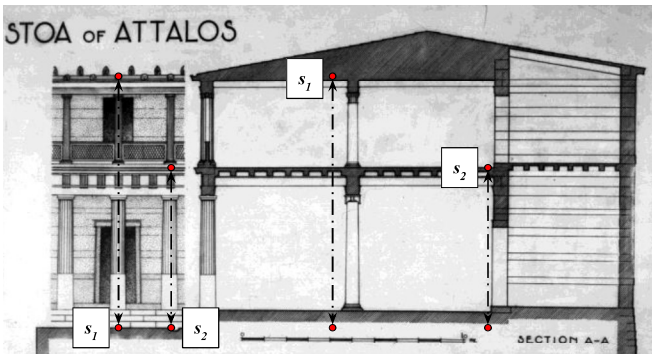
For comparison, we also tested the performance of Single View Metrology (SVM) as described in (Criminisi et al., 2000). The results indicate that our method outperforms SVM, when applied both on the controlled and Monte Carlo experimental setups. The performance of SVM in the controlled and Monte Carlo experiment was 4.29% ± 1.96% and 7.32% ± 6.81%, respectively. These results indicate that the proposed methodology is more accurate for measuring objects using a single image while maintaining a more stable performance since the standard deviation (*Std.*) is significantly lower. The advantage of the proposed methodology over SVM, was also assessed by using the two paired *t*-test as statistically significant ($p < 0.001$ at a significance level of $a = 0.05$). Additionally, when the performance is analyzed in detail, as presented in Tables 2, 3, it can be observed, that both the weighted and non-weighted version of the proposed methodology are more versatile since their performance in terms of MAPE is more robust regardless the view angle and object size.

Table 5.5. Average MAPE per object on natural image dataset

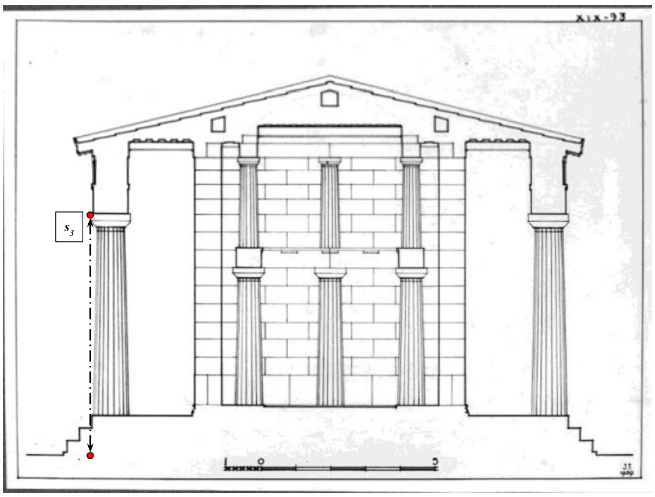
Objects	Methods					
	Weighted		*Non-Weighted*		*SVM* (Criminisi et al., 2000)	
	MAPE	Std.	MAPE	Std.	MAPE	Std.
1	2.90	1.76	3.81	2.71	3.79	2.85
2	3.23	1.7	4.79	3.23	4.46	3.35
3	3.18	1.98	4.98	3.2	4.66	3.3
4	3.08	1.54	4.88	3.17	4.94	3.18
5	2.57	1.37	5.38	2.54	5.39	2.41
6	2.94	2.17	4.18	2.98	4.17	3.01
7	4.14	3.56	4.59	1.71	4.25	1.83
8	3.82	2.01	7.74	2.61	7.33	1.85
9	2.33	1.77	6.27	2.33	5.67	3.31
Average	3.13	0.51	5.18	1.19	4.96	1.10

The dataset that was constructed with natural images (not simulated), consisted of 9 different parallelepipedal objects, with their side dimension ranging from 7.8 cm to 53 cm. The objects were captured from different distances and viewing angles with respect to the camera. Each different position was marked on a surface where both the objects and the camera were placed on throughout the experiment (Figure 5.3). The positions were measured using a measuring tape. All the objects were placed on the same positions and captured using the Intel RealSense® D435. The intrinsic parameters of the Intel RealSense® D435 RGB sensor are provided by the manufacturer. RealSense® D435 RGB sensor has a focal length of f_{pixels} = (613.71, 613.79) and a principal point c = (322.23, 240.52). The image resolution was 640 × 480 pixels.

During the experiment, the camera was mounted on a tripod, on a height of 12 cm. The camera was leveled so that its *X*-*Z* plane defined by its *X*- and *Z*-axis be approximately parallel to the ground plane. The bounding boxes were extracted by a human. In addition, to cross validated the distances between the camera and object according to the annotated marker on the surface, the distances also have been measured and logged using the Intel RealSense® D435 stereoscopic capabilities. The MAPE that was achieved on the natural image dataset, by the proposed methodology, accounting for all the objects measured was 3.13% ± 0.51% (Table 5.5). Measurements using both the proposed method without the weights and the SVM approach were performed. The non-weighted approach of the proposed method achieved a MAPE of 5.18% ± 1.19%, providing more evidence that the weighted aggregation approach has a significant impact to the accuracy of the measurements. The SVM method, provided a MAPE of 4.96% ± 1.10%. The proposed method, with the weighted aggregation step had the highest measurement accuracy while achieving a small *Std.*, indicating a more stable performance overall. A summary of the measurement results per object is presented in Table 5.5.



(a)



(b)

Figure 5.4. Scaled plans of the monuments used in the evaluation process, marked with the linear segments s1, s2 and s3. (a) Stoa of Attalos, (b) Temple of Hephaestus.

To further evaluate the performance and robustness of the proposed single image measurement methodology, we evaluated it by measuring objects illustrated in natural images captured *in-the-wild*. In detail we chose two well-known landmarks located in the Ancient Agora of Athens, Greece, namely the Stoa of Attalos and the Temple of Hephaestus illustrated in Figure 5.5 (a, b) and Figure 5.5 (c) respectively. The dimensions of these landmarks are widely known and documented. In addition, architectural plans of these monuments along with the respective scale of the plans (Figure 5.4 (a, b)) are provided by the American School of Classical Studies in Athens (Classical Studies, n.d.).

Regarding the Stoa of Attalos, the measurements that were performed were along the height of the monument. Two vertical linear segments were chosen to be measured, one spanning along the total height of the monument (s_1) and one (s_2) defining the height of the first floor. The height of the Stoa of Attalos is reported to be 11.42m (s_1) (Bernard & Pike, 2015) which was used as ground truth. The size of (s_2) was measured to be 6.97m by exploiting the scale provided in the respective plan of the monument (Figure 5.4 (a)). Additionally, s_2 was measured from two different perspectives (Figure 5.5 (a, b) since the spatial layout of the broader area of the Stoa of Attalos was appropriate to capture the monument in that way while maintaining s_2 in the image. Regarding the Temple of Hephaestus, we measured the linear segment depicted in Figure 5.5 (c) (s_3). The real size of s_3 was calculated to be 6.96m by using the respective plan of the temple, which similarly to Stoa of Attalos' plans, is scaled. The segments of s_1, s_2, s_3 were measured along each column where both the column and each segment were clearly visible in the image. In detail, regarding the Stoa of Attalos, s_1 was sampled along 10 different columns depicted in Figure 5.5 (a), whereas s_2 was sampled along 18 different columns from both Figure 5.5 (a, b). The segment s_3 corresponding to Temple of Hephaestus, was sampled along 4 different collumns depicted in Figure 5.5 (c).

Table 5.6. Comparison of average measurements MAPE per segment on images captured in-the-wild

Segments	Methods							
	Weighted		*Non-Weighted*		*SVM* (Criminisi et al., 2000)		*RealSense*	
	MAPE	Std.	MAPE	Std.	MAPE	Std.	MAPE	Std.
s_1	5.33	4.26	8.29	6.59	9.54	7.86	57.03	22.75
s_2	4.15	3.03	4.59	3.77	4.43	3.64	36.28	33.59
s_3	20.89	14.54	28.15	13.50	27.90	13.59	62.15	43.01
Average	8.63	10.28	11.41	12.45	11.57	12.50	47.93	34.33

(a)

(b)

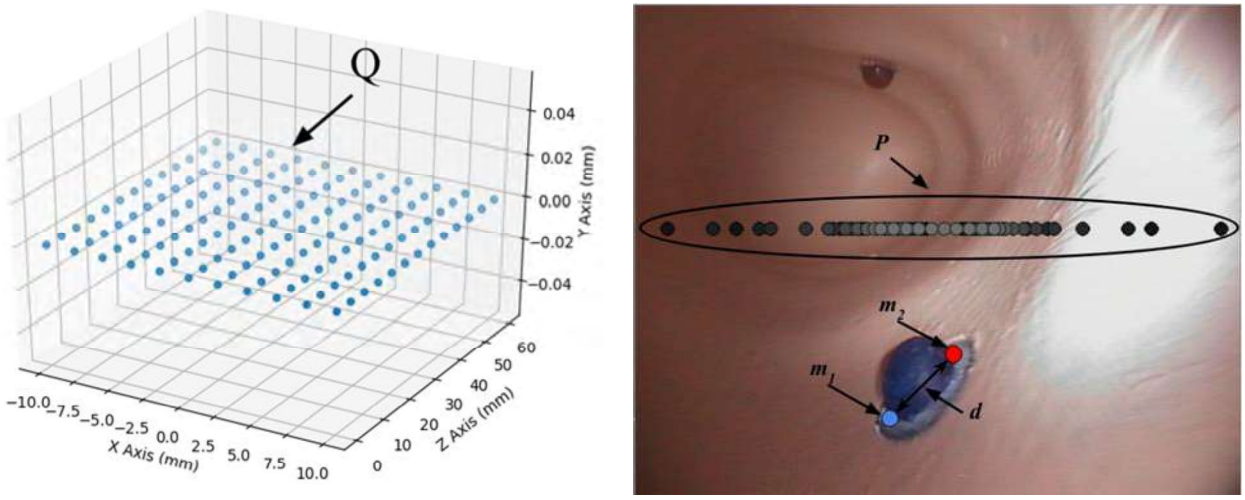
Figure 5.5. Images illustrate the monuments that were measured marked with the linear segments s1, s2 and s3. (a, b) Stoa of Attalos (c) Temple of Hephaestus.

The images that were used for the evaluation were captured using the Intel RealSense® D435 stereo camera. The camera was held on a height of 1.90m by a user; however, the exact positioning of the camera was unknown, *i.e.*, the degrees of roll and tilt. For the SVM method the horizon line was manually extracted. Furthermore, we included in the comparison, the measurement results using the depth maps provided by the Intel RealSense® D435.

The results of the *in-the-wild* evaluation are summarized in the Table 5.6. The proposed methodology achieved an average MAPE of 8.63% ± 10.28%, which was the lowest amongst all. The weights that were used were estimated using the Webots simulation platform. The non-weighted approach achieved the second lower error, scoring a MAPE of 11.41% ± 12.45%. The average error achieved by the SVM method, and the stereo (RealSense) approach were 11.57% ± 12.50% and 47.93% ± 34.33% respectively. The high error that the stereo approach, using the depth maps deriving from the RealSense camera, could be attributed to the long distance between the camera and the segments that were measured. D400 series depth cameras are documented to give the most precise depth data in short distances. This is due to the non-linear

scaling of the depth prediction error along the depth of the scene (Grunnet-Jepsen, Sweetser, & Woodfill, 2018).

5.2 In-Vivo Single Image Measurements



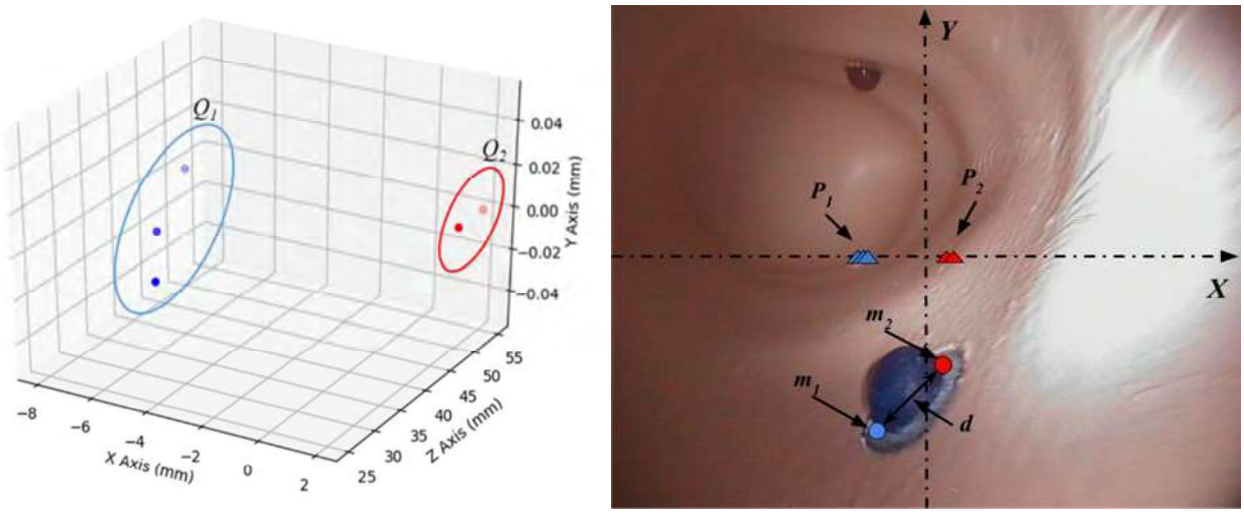
(a) (b)

Figure 5.6. Example 3D-to-2D point correspondences. (a) 3D representation of the grid with the points of set Q. (b) The projection P of Q on the image plane with respect to the selected points m_1 and m_2. (c) The linear segment d to be measured, defined by m_1 and m_2. The grey-level intensities of the points in P represent depth, i.e., points lying deeper appear with a higher intensity.

As a first step in the proposed methodology, two points of an object of interest, *e.g.*, a lesion in the GI tract, are selected in an endoscopic image. These points define a linear segment to be measured. Then, a set *Q* of 3D points forming a grid is defined (Figure 5.6 (a)). The 3D point grid expands only on *XZ* plane as it is sufficient for efficient implementation of the proposed methodology. Assuming that the intrinsic parameters of the camera are known, all the 3D points of *Q* are projected on the 2D image plane based on the pinhole camera model to a set *P* of 2D points (Figure 5.6 (b)). Thus, 3D-to-2D correspondences are obtained. For visualization purposes Figure 5.6 (b) represents the points of *P* with different grey-level intensities to indicate that they originate from points of *Q* obtained from different depths set *Q*. This visualization does not show all the points of *Q* as projected in *P* because the points are overlapping in the 2D space.

Also, the points in Figure 5.6 (b) are not uniformly distributed because of the conic field of view of the camera. As a next step, the selected points of the 2D image plane are matched with the 2D

projections on this plane of the 3D points based on their Euclidean distances. A problem is that some points that are close to each other, on the image plane, may have different distances in the 3D space. To tackle this problem, we introduce an additional step for the depth estimation. Therefore, when the depth value (*z* coordinate) of the corresponding 3D point is determined, it is



(a) (b)

Figure 5.7. Graphical representation of the sets Q_1 and Q_2 with their respective 2D correspondences P_1 and P_2. (a) 3D representation of Q_1 and Q_2. (b) Example of sets P_1 and P_2 with respect to m_1 and m_2.

used to perform the size measurement by exploiting the geometric properties of the pinhole camera model.

Let us consider a user-defined linear segment d (Figure 5.6) on the object of interest to be measured given two 2D points $m_1 = (u_1, v_1)^T$, $m_2 = (u_2, v_2)^T$. To be able to measure the size of this linear segment, i.e., the distance between these two points in the 3D space, the intrinsic parameters of the camera are needed. Then we need to estimate the depth of these points and project them to the 3D space. The intrinsic parameters of the camera can be recovered by a calibration procedure (Zhengyou Zhang, 1999). Once the intrinsic parameters of the camera are recovered, namely the focal length $f_p = (f_x, f_y)$ in pixel units, the principal point $c = (u_0, v_0)$ and the distortion correction parameters, we can artificially project any known 3D point, onto the 2D image plane.

The generation of the 3D point cloud is similar to the one presented in section 5.1. Let Q be a set of 3D points $q = (x, y, z)^T$, where $x, y, z \in \mathbb{R}$ are coordinates in the XYZ cartesian coordinate system (Figure 5.6). The X and Y axes are along the width and the height of the GI tract while the Z-axis is along its depth. Each point q, in our 3D point set Q, satisfies the following conditions:

$$mod(x, s_x) = 0, -a \leq x \leq a, \quad (5.17)$$

$$y = 0 \quad (5.18)$$

$$mod(z, s_z) = 0, 1 \leq z \leq b \quad (5.19)$$

where $a, b \in \mathbb{R}$ are variables that delimit the maximum and minimum values of x and z coordinates, and $s_x, s_y \in \mathbb{R}$ are values setting the distance between the consecutive coordinates x and z of each 3D point, and *mod* represents the modulo operation. Thus, set Q can be considered as a 3D point cloud on the XZ plane (Figure 5.6 (a)). Since set Q is defined, and the intrinsic parameters of the camera are known, any 3D point $q \in Q$, can be projected on the 2D image plane. Based on the

pinhole camera model, a projection of a 3D point $q = (x, y, z)^T$ to the 2D image plane can be described as follows (Heikkila & Silvén, 1997):

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{z}\begin{pmatrix} f_x x \\ f_y y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{5.20}$$

To create 3D-to-2D correspondences, we apply Eq. (5.20) on all the points of set Q. As a result, a new set of 2D points P where each point $p \in P$, corresponds to a known 3D point q. To find the depth values of points m_1 and m_2, we perform a matching procedure among the 2D points of the set P with points m_1 and m_2. The matching is based on the Euclidean distance between the points of the set P and the selected points m_1 and m_2.

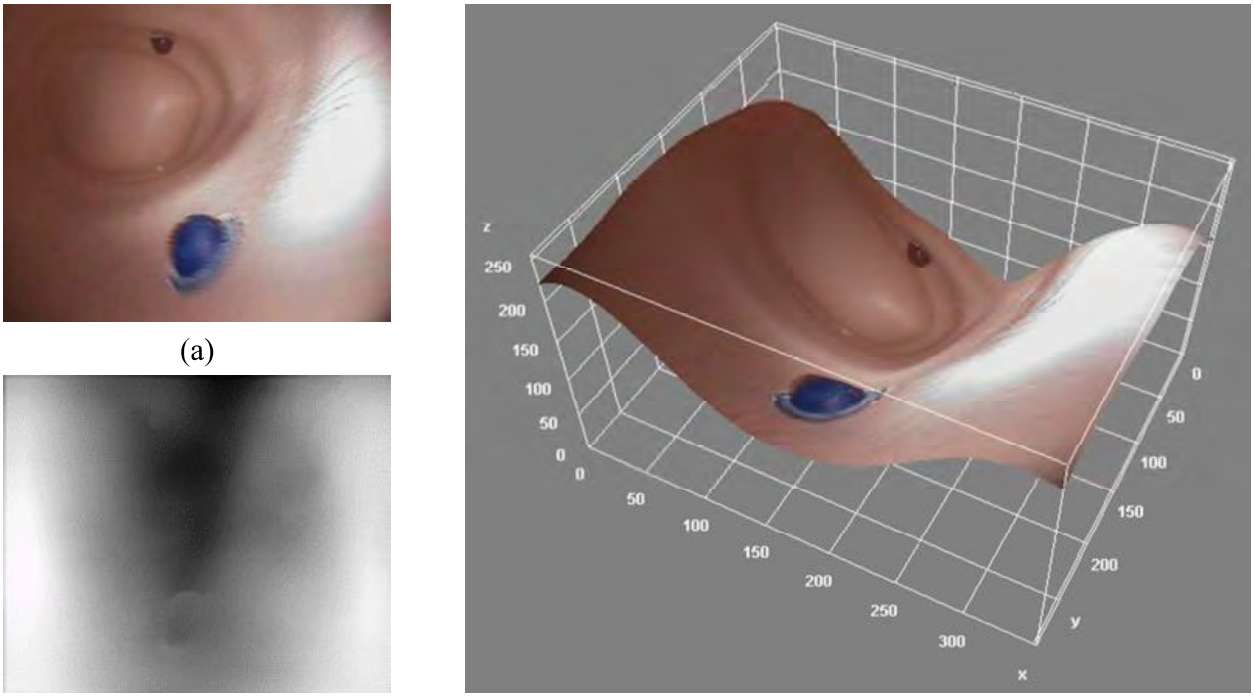
From Eq. (5.20), it can be easily derived that the 3D points with the same x and z coordinates in the 3D space, will also have the same 2D pixel coordinate u in the 2D image space. By taking into account the conditions (5.17)-(5.19), and Eq. (5.20), all the 2D points of set P will have the same pixel coordinates v but different pixel coordinates u. Each point p of P that share similar or the same pixel coordinates u with points m_1 and m_2 may as well have the same or similar z values in the 3D space. To find these points p, that have similar pixel coordinates u with points m_1 and m_2, we calculate the Euclidean distances among them, and we choose a subset of these points p that have a Euclidean distance from m_1 and m_2, that is below a threshold t. This step results into two sets of points, P_1 and P_2 that can be formally expressed as follows:

$$P_1 = \{p \in P: \|p - m_1\|_2 \leq t\} \tag{5.21}$$
$$P_2 = \{p \in P: \|p - m_2\|_2 \leq t\} \tag{5.22}$$

with $\| \quad \|_2$ denoting the 2D Euclidean distance (norm). The value of threshold t controls the number of point matches between p and m_1, m_2. Larger values of t result in more point matches; however, the quality of the matches deteriorates. Smaller values of t result in higher quality matches, but it should not be too low, depending on the density of the grid, as it may result in no matches.

All points of P_1 and P_2 respectively, are basically projections of known 3D points q in the 2D image plane; therefore, their corresponding x and z coordinates, are known. Some of these coordinates may be close or the same to those of m_1 and m_2. However, due to the nature of the projection, the points of P_1 and P_2, may have different depth values z even if they have similar pixel coordinates $(u, v)^T$ in the 2D image plane. For this purpose, a depth approximation approach is considered for the refinement of the 2D to 3D correspondences.

Let Q_1 and Q_2 be the sets of all 3D points that correspond to the 2D points of P_1 and P_2 respectively (Figure 5.7). The 3D projections of m_1 and m_2 from the image plane to the 3D space are assumed to be on the same depth, and thus, both m_1 and m_2 will share the same z value. To estimate the z value of m_1 and m_2 using the corresponding sets Q_1 and Q_2, we introduce an intermediate machine learning assisted step by using a pre-trained Convolutional Neural Network (CNN) that have been



(a) (b) (c)

Figure 5.8. Example of a generated depth map Dm using a CNN [36]. (a) Endoscopic image used as input to the CNN. (b) The output of the CNN. (c) 3D representation of the endoscopic image using D_m for the representation of depth.

previously proposed for depth map estimation from monocular RGB images (Alhashim & Wonka, 2018). The CNN has an auto-encoder architecture, with skip connections. The encoder of the network is a DenseNet-169 (G. Huang et al., 2017). The decoder consists of bilinear ×2 up-sampling layers followed by two convolutional layers. Since ANNs are considered as function approximators, the proposed CNN model, using an encoder-decoder architecture can be formally expressed as follows:

$$\varepsilon : I \rightarrow \Lambda \tag{5.23}$$

$$\delta : \Lambda \rightarrow D \tag{5.24}$$

$$\varepsilon, \delta : argmin_{\varepsilon,\delta}\left(\frac{\lambda}{N}\left\|D_m^{gt} - (\varepsilon \circ \delta)I_{RGB}\right\|_1 + \frac{1}{N}\left\|\nabla D_m^{gt} - \nabla(\varepsilon \circ \delta)I_{RGB}\right\|_1 + \frac{1 - SSIM(D_m^{gt}, (\varepsilon \circ \delta)I_{RGB})}{2}\right) \tag{5.25}$$

The encoder function is denoted with ε, and its purpose is to map the input data of space I which consists of all RGB images I_{RGB}, to a latent space Λ. Then, the decoder function, denoted as δ, has the task to predict the ground truth depth map D_m^{gt} of space D, that corresponds to the scene of I_{RGB}, using an instance of latent space Λ. Both the encoder and decoder networks are trained to minimize Eq. (5.25).

For the needs of this methodology, the CNN receives as input an RGB endoscopic image and generates a normalized depth map D_m expressing the relative distance of the object to be measured, from the camera. An example is illustrated in Figure 5.8. The output of the CNN is relative and approximate; therefore, it cannot be considered directly for absolute depth estimation. It is considered only as a tool for the selection of appropriate points from sets Q_1 and Q_2, which include 3D points more accurately estimated by the calibration procedure described above.

To select points from sets Q_1 and Q_2 that correspond to m_1 and m_2, the values of D_m are quantized into three levels that intuitively express *short*, *medium* and *long* distances from the camera (without quantizing D_m values result in inferior performance as it can be confirmed by the results presented below). If an object to be measured in the RGB image belongs to a region of *short*, *medium* or *long* distance, the *minimum*, *mean* or *maximum z* value from both the sets Q_1 and Q_2, respectively, is selected as the most appropriate depth value of m_1 and m_2 points.

Experiments and Results

For the experimental evaluation of this methodology an experimental workbench was used. That experimental setup aims at mimicking a colonoscopic scenario through controlled and repeatable instrumentations. A conventional endoscope was manually navigated into a rigid colonic phantom, reconstructed from a real colon computed tomography (CT), selected from the Cancer Imaging Archive dataset (Prior et al., 2017). Due to the use of non-deformable material for the physical realization of the phantom, *i.e.*, ABS polymer, the diameter and curvature angles of the reconstructed colon have been slightly scaled-up to ensure the progress of the colonoscope along the entire path of the phantom. Several colored targets were installed and fixed in calibrated positions, internally along the colonic tract. The targets aim at reproducing landmarks, characterized by different shapes and sizes: parallelepiped-shaped (6.36 mm × 6.36 mm × 4.5 mm), hemisphere-shaped (4.5 mm and 1 mm). An electromagnetic tracking system (EMTS), positioned under the colonic phantom, was used to track both the position of the endoscope and of the rigid phantom with two independent sensors, inserted and fixed at the tip of the operating channel and attached to the colonic phantom, respectively.

The evaluation of the proposed methodology was achieved by performing size measurements, with respect to the length of the target object to be measured. The measurements were performed on all the available targets, *i.e.*, 12 hemispherical objects with a diameter of 4.5mm and 1mm, and 6 parallelepipedal with a square base of 6.36mm, from various distances and perspectives, in 218 different video frames. To define the linear segments on the objects, an endoscopist annotated the points on the images, as done in clinical practice. For the estimation of the intrinsic parameters of our camera endoscope, we used Zhang's (Zhengyou Zhang, 1999) calibration methodology, as implemented by Bouget (Bouguet, 2004), which includes pre-processing for lens distortion correction. For the calibration process, we used 40 images of a checkerboard pattern. The focal length was estimated to be f_p = (402.36, 435.59) and the principal point c = (359.11, 263.22).

Table 5.7. Comparative results of measurements on hemispherical objects (in mm) obtained using the proposed and the EMTS methods.

Relative Distance	Proposed 3-level D_m	Proposed Full D_m	EMTS
Short	1.17 ± 0.68	1.45 ± 1.31	1.12 ± 0.76
Medium	0.70 ± 0.20	3.07 ± 0.50	4.79 ± 0.47
Long	0.93 ± 0.69	2.77 ± 1.10	3.54 ± 2.61
Average	0.98 ± 0.66	2.36 ± 1.29	2.89 ± 2.34

Table 5.8. Comparative results of measurements on paral. objects (in mm) obtained suing the proposed and the EMTS methods.

Relative Distance	Proposed 3-level D_m	Proposed Full D_m	EMTS
Short	2.90 ± 1.50	3.11 ± 1.65	1.59 ± 1.39
Medium	1.28 ± 0.94	3.15 ± 0.71	4.61 ± 2.62
Long	1.97 ± 0.86	2.93 ± 0.60	2.20 ± 1.65
Average	2.05 ± 1.10	3.06 ± 0.98	2.80 ± 1.89

Table 5.9. Comparative results of measurements on 1mm objects (in mm) obtained suing the proposed and the EMTS methods.

Relative Distance	Proposed 3-level D_m	Proposed Full D_m	EMTS
Short	0.28 ± 0.19	0.62 ± 0.35	0.21 ± 0.09

By following the methodology, described in above, we measured the length of the selected objects. The performance of our method was measured by the Mean Absolute Error (MAE) between the estimated and the actual size of the measured objects. From the 6 hemispherical objects appearing in the dataset, 2 were identified as in *short*, 1 as in *medium* and 3 as in *long* distance from the camera. From the 6 parallelepipedal objects appearing in the dataset, 2 were identified as in *short*, 2 as in *medium* and 2 as in *long* distance from the camera. Regarding the 1mm-hemispherical objects, they were all identified in *short* distances from the camera, since they were not clearly visible from longer distances.

The average MAE achieved on the 4.5mm-hemispherical objects and on the parallelepipedal objects, using depth maps quantized in *short*, *medium* and *long* distance, was 0.98mm ± 0.66mm and 2.05mm ± 1.10mm respectively. The average MAE obtained with the same method on the 1mm-hemispherical objects was 0.28mm ± 0.19mm. The results per distance are presented in Table 5.7-Table 5.9. In comparison we also provide the results of the proposed methodology using the full range of the depth values in D_m instead of using the 3-level quantized D_m. The results validate that the MAE in the latter case is larger, in detail, 2.36mm ± 1.29mm for the 4.5mm-hemispherical objects, 3.06mm ± 0.98mm for the parallelepipedal objects and 0.62mm ± 0.35mm for the 1mm-hemispherical objects.

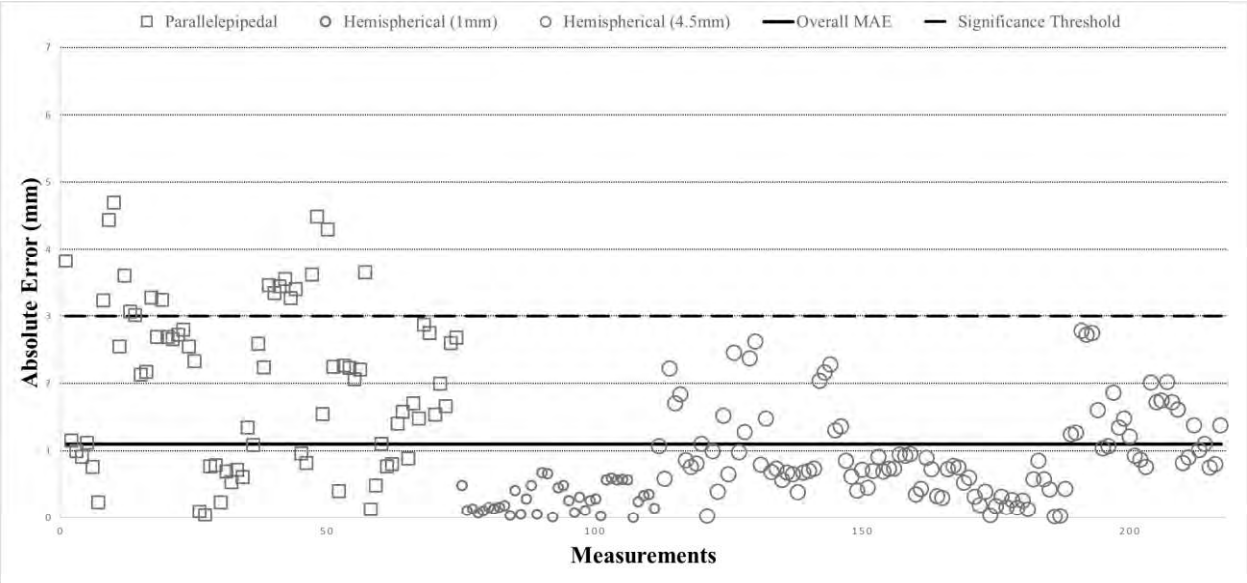


Figure 5.9. Size measurement MAE per measured object.

Another comparison was performed using the measurements of the targets using the *z* depth values from the object to the camera, provided by the EMTS. Using the EMTS *z* values for the measurement of the 4.5mm- hemispherical, the parallelepipedal and the 1mm-hemispherical objects, the average MAE was 2.89mm ± 2.34mm, 2.80mm ± 1.89mm and 0.21mm ± 0.09mm, respectively. It is worth noting that the error using the EMTS method is higher than the one obtained by the proposed methodology with an exception to the measurements of the 1mm objects where it achieves better results. This is reasonable since all of these objects were captured in short distances and the EMTS performs better for short distances, as it can be observed in Table 5.7-Table 5.8.

An indication of the accuracy of the proposed size measurement approach can be obtained by following the significance level criteria proposed in (Schoen, Gerber, & Margulies, 1997). In that study a measurement is considered inaccurate when the MAE is greater than 3mm. As it can be seen in Figure 5.9, the majority of our measurements are below the 3mm threshold. In Fig. 6, the solid black line, indicates the respective MAE among all measurements. The dashed line represents the 3mm error significance threshold.

5.3 Discussion and Future Work

In section 5.1, a novel, single image, uncertainty-aware measurement methodology is proposed. The proposed method is based on the establishment 3D-to-2D correspondences by projecting virtual 3D points to the 2D image plane. These correspondences are exploited to perform measurements on an image, in metric units, without the need of an image sequence, or any depth estimation CNN or any sensor-based framework. Therefore, it can be considered as a solution for VMs with low-cost camera-enabled devices, such as conventional smartphones, robotic systems,

unmanned vehicles, and assistive systems, which promises impactful applications(Oh & Han, 2021; Yan, He, Basiri, & Hancock, 2019).

Regarding the performance of the proposed (weighted) methodology, it outperforms its non-weighted counterpart and SVM (Criminisi et al., 2000). A detailed comparison between the proposed and other methods was presented in Table 5.2-Table 5.5. The comparative results were conducted by using a dataset consisted of synthetic images and natural images depicting objects of various dimensions. Furthermore, we evaluated and compared the performance of the proposed method on images illustrating monuments of known dimensions *in-the-wild*. The proposed method outperformed all other methods including a stereo approach using the Intel RealSense® D435 camera (Table 5.6).

The proposed method considers the uncertainty factor of the initial depth estimation step. Probable depths are considered near the initial depth value which are aggregated using a trainable step providing a more accurate depth estimate. A limitation of the proposed methodology relied to the generation of virtual 3D point set Q. The generation of Q is constrained by the parameters a, b that introduce additional complexity in comparison to SVM. However, these parameters can be easily determined in a simulated environment as demonstrated in the experiments and results of section 5.1. The results provided by the proposed methodology on datasets consisting of synthetic and natural images were consistent by utilizing the parameters initialized in the simulated environment. Furthermore, in contrast to SVM that is only capable of height estimation, the proposed method, estimates the depth of an object which enables its measurement along all dimensions. In contrast to SVM, the proposed method does not require any information regarding the horizon line; hence it is more versatile to changes in the positioning of the camera and in environments where the horizon line detection process can fail. Moreover, the depth estimation approach, is far less complex and computationally intensive in comparison to deep learning models. The depth information provided by our method, can be exploited to various applications, such as obstacle detection and risk assessment with respect to its distance, viewing angle and size eliminating the need for additional specialized sensors, such as stereo camera rigs that are usually employed in such cases as it was described in section 5.1.

In section 5.2, we investigated a methodology for *in-vivo* visual measurement of objects, *e.g.*, lesions, in endoscopic images. Previous relevant methodologies are mainly based on the use of reference objects, or on the use of multiple consecutive video frames of the objects in order to perform measurements (Dimitris K Iakovidis et al., 2019). Unlike these methodologies, the proposed one requires only a single image of the object of interest. Also, the evaluation of the proposed methodology was performed using a more realistic experimental setup, using a 3D printed colonic phantom and freely moving camera. The most relevant previous approaches (George Dimas, Iakovidis, Karargyris, Ciuti, & Anastasios, 2017; George Dimas, Spyrou, Iakovidis, & Koulaouzidis, 2017b; Dimitris K Iakovidis et al., 2019) are based on straight (unfolded) bowel phantoms with 1-DoF camera motion.

The hemispherical objects used as measurement targets were constructed in such way to represent sessile polyps (Arebi et al., 2007). Polyps of this type are elevated from the adjacent mucosa without a clear stalk (Paris Workshop, 2003). Other types of polyps include pedunculated, hyperplasia, and inflammatory with different geometrical structures. However, instead of trying to model all the different types of polyps, which can be a complicated task considering their diversity, we have included the parallelepipedal shapes to demonstrate the capability of the proposed methodology to measure objects of different shapes. The application of the proposed methodology for the measurement of diverse specimens, can be performed in the context of a medical study validating the results of the research presented in section 5.2.

The MAE of the most recent relevant methodology (Dimitris K Iakovidis et al., 2019), was 0.19mm ± 0.17mm with respect to the size of the measured objects, which were circular pins of 0.95mm diameter. These pins were attached on a 30cm straight small bowel phantom, and the measurements were performed using a wireless capsule endoscope. By applying the proposed methodology on that dataset, the obtained MAE was 0.18mm ± 0.10mm, which is marginally lower. It is worth noting that this result was achieved using only a single image, whereas the methodology presented in (Dimitris K Iakovidis et al., 2019), requires multiple images to perform the size measurements. Also, the EMTS approach resulted in a MAE among all measurements of 1.96mm ± 1.52mm which is significantly larger than the respective MAE of our method which is 1.10mm ± 0.89 mm. Besides the performance of the EMTS, it requires equipment that would hardly be used during an actual WCE examination.

An intermediate step of the proposed methodology uses a CNN that predicts the relative distance from the camera given a monocular RGB image. However, this CNN is pretrained on non-endoscopic data; therefore, the quality and accuracy of the depth map may not be optimal. To cope with this issue a rough, intuitive, approximation distance is considered, in terms of *short*, *medium*, and *long* semantic descriptors.

Overall, a significant advantage of the proposed method is that it requires only one image for accurate size-measurement of an object, whereas other methods require multiple images (Dimitris K Iakovidis et al., 2019), reference objects (Kume, Watanabe, Yoshikawa, & Harada, 2014)(Kaz, Anwar, O'Neill, & Dominitz, 2016), lasers and structured light (Goldstein, Segol, Gross, Jacob, & Siersema, 2017)(Visentini-Scarzanella et al., 2018). Disadvantages of the proposed method include its dependency on the density of the point grid, which is manually defined by the parameters $\underline{s_x}$ and s_z, and its dependency on the quality of the depth map estimation.

Perspectives for future research include automated approaches for the estimation of parameters $\underline{s_x}$ and s_z, and methods for more accurate depth estimation method, which could further improve the performance of the proposed method. However, this would require the construction of a training set of endoscopic images with ground truth depth maps. Such depth maps could be obtained with special setups, *e.g.,* stereoscopic imaging. As it was noted by (Oka et al., 2014), the accuracy of the measurements may be affected by the orientation of the camera towards the target object. In this study, we have not taken into consideration the angle of view towards the objects to be

measured. This could be another source of uncertainty, which we intend to investigate in a future study, with an enhanced experimental setup enabling the assessment of the angle of view. Other future improvements of the proposed approach may include considerations of other uncertainty factors, such as the determination of the points selected by the physician.

It is worth noting that in an extensive review study performed by (Dimitris K Iakovidis & Koulaouzidis, 2015), indicates that the topic of endoscopic size measurements has not been sufficiently investigated, although it can be significant for malignancy assessment. The proposed methodology contributes towards filling the gap of performing *in-vivo* measurements based on computer vision, without the use of any external reference.

The proposed methodology is valuable in a variety of applications well beyond GI endoscopy. It is applicable on any endoscopic imaging modality either in the medical domain, *e.g.,* laparoscopy, colposcopy, etc., or in other domains, *e.g.,* pipe inspection, and it can be extended to cover also non-endoscopic visual size measurements.

As a future work we intend to further improve both methodologies mainly with respect to their parametrization and its dependence on the height knowledge. The parametrization can be automated according to the camera parameters and the scale of the objects that are needed to be measured.

Chapter 6
Deep Learning in 3D Modeling

Another interesting subject is the investigation of the capacity of a DL model to perceive and represent complex 3D structures. In the last few decades, three-dimensional (3D) models of complex tissues, such as human organs, have been developed for a wide range of applications in the medical and biomedical engineering fields. Using such 3D models, the function of an organ under different normal or pathological conditions can be simulated using *in silico* models referred to as digital twins (P. G. Kalozoumis, Marino, Carniel, & Iakovidis, 2022). To enhance both product efficacy and patient safety, biomedical products, such as medical devices, are extensively tested prior to commercialization, usually with extensive animal and human studies. *In silico* clinical trials (ISCT) can provide solutions to these issues by partially replacing or complementing clinical trials for testing medical devices and drugs. Nevertheless, the accuracy of the simulation outcome depends highly on the fidelity of the 3D model characteristics. A mesh representation of the geometric structure of a target object is usually extracted from one or more images of that object, *e.g.*, magnetic resonance (MR) or computed tomography (CT) images. However, the resolution of the acquired images may be insufficient, resulting in coarse 3D models with low geometric accuracy, which can compromise the simulated outcome.

Implicit representations have exhibited a capability to express shapes in the form of continuous functions with the use of artificial neural networks (ANNs) (Z. Chen & Zhang, 2019; Chibane, Pons-Moll, & others, 2020; Mescheder, Oechsle, Niemeyer, Nowozin, & Geiger, 2019; Park, Florence, Straub, Newcombe, & Lovegrove, 2019) or convolutional neural networks (CNNs) (S. Peng, Niemeyer, Mescheder, Pollefeys, & Geiger, 2020). These networks, known as implicit neural representations (INRs), are tasked to approximate implicit functions based on raw data, point clouds (PCs), or latent codes with or without supervision (Mescheder et al., 2019; Park et al., 2019; Sitzmann, Martel, Bergman, Lindell, & Wetzstein, 2020). Signed distance functions (SDFs) have been recently utilized in INRs to infer different geometries (Chabra et al., 2020; Ma, Han, Liu, & Zwicker, 2020; Park et al., 2019; Sitzmann et al., 2020). However, many of these

approaches require 3D supervision and cannot solely use raw PC data. This problem has been tackled by using the Eikonal equation to approximate SDFs (Gropp, Yariv, Haim, Atzmon, & Lipman, 2020; Sitzmann et al., 2020).

Regarding methods related to the 3D reconstruction of complex human tissue structures, early approaches have focused on conventional methods such as structure from motion (Hu, Penney, Edwards, Figl, & Hawkes, 2007). More recently, generative models for 3D organ shape reconstruction have been proposed using autoencoder architectures (Balashova et al., 2019; Z. Wang et al., 2020). The related research has produced promising results; however, it has mainly focused on learning-based models requiring 3D supervision and computationally demanding neural network architectures. On the other hand, INR approaches have been applied only on high-resolution image reconstruction (Reed et al., 2021; Wu et al., 2021).

In section 6.1, we leverage the capacity of multilayer perceptrons (MLPs) to learn continuous INRs for the reconstruction of high-quality 3D models of the gastrointestinal (GI) tract from initially coarse representations of these models. More specifically, an MLP-based INR is proposed, where the neural network has a novel activation function, called hereinafter WaveShaping (WS). The WS function improves the performance of recently-proposed periodic activation functions that have been used in the Sinusoidal Representation Networks (SIREN) (Sitzmann et al., 2020). Moreover, the form of its first derivative benefits the training process, enabling the network to provide better reconstruction results during inference. Thus, coarse 3D models of the GI tract can be properly restored to be used in simulation configurations. To the best of our knowledge this is the first time that INRs have been used to reconstruct 3D models of the GI tract from coarse 3D representations.

6.1 3D Reconstruction Using Implicit Neural Representations

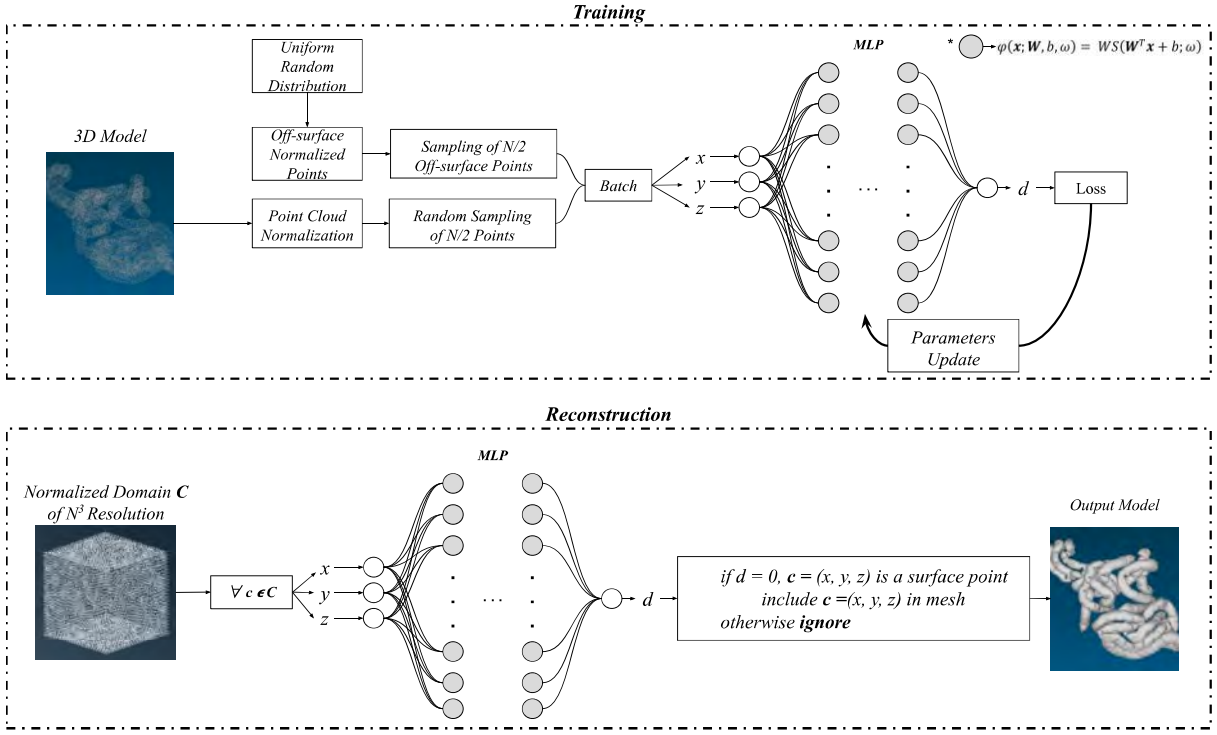


Figure 6.1. Overview of the INR approach for 3D model refinement and restoration.

The proposed methodology aims at reconstructing 3D coarsely retrieved models of the GI tract by employing an MLP network tasked to learn an implicit continuous representation of that model (Figure 6.1). The proposed methodology does not require training on large datasets since it focuses on a single 3D model. The MLP network, utilizes the proposed WS activation function to learn an SDF (Osher & Fedkiw, 2003), which describes efficiently the 3D model that it aims to reconstruct. The MLP receives as input a point $u = (x, y, z)^T$ of a 3D model, and it outputs a value approximating the respective SDF response. That value describes the distance of a point from the surface of the 3D model. After the training process, the model is capable of reconstructing the 3D model, which was initially coarse, at a higher resolution by predicting through inference if a point in the defined 3D space belongs to the surface of the model, *i.e.*, its distance from the surface is 0.

Let us consider $p = (x, y, z)^T$, $x, y, z \in (-1, 1)$ as a point of a normalized PC P, with 3D points lying on the surface of a modeled object. This surface is described by an SDF iso-surface $s(\cdot)$, s: $\mathbb{R}^3 \rightarrow \mathbb{R}$ (Osher & Fedkiw, 2003), which encodes the surface of the 3D model as the signed distance of a point p to the closest surface. The points that are outside and inside of the object's surface are denoted by a positive and negative distance d, respectively. A response $d = 0$ denotes a surface point p of the 3D model. An SDF that describes a particular 3D model can be approximated by a neural network. Therefore, a 3D model is implicitly represented by the weights of the MLP.

An SDF describing a 3D model of the GI tract can be approximated by an appropriately trained MLP $g(\cdot\,;\theta)$ parametrized by θ. The MLP $g(\cdot\,;\theta)$ should learn how the SDF responds for different points p that are on and off the surface of the model. Thus, the training dataset should be arranged to contain the points p of the PC P that lie on the surface of the model and random normalized samples of points $q = (x, y, z)$ that reside on the rest of the domain. These points p can be sampled from a random uniform distribution consisting of x, y, and z coordinates, within the range of [-1, 1]. Each training batch N is composed of the points p and q with $N/2$ points each.

In our approach the loss function proposed in (Sitzmann et al., 2020) was used to train MLP $g(\cdot\,;\theta)$. Once g is trained it can be used to reconstruct an initially coarse model of the GI tract at higher resolutions. Let us consider a cubic normalized PC K with a density n^3 and points $k = (x, y, z)^T$, $x, y, z \in [-1, 1]$. By inferring SDF values $\forall\, k \in K$ using g, the coarse model that was used for training can be reconstructed with higher detail by examining the responses of the network. Zero and non-zero responses denote that a point k resides on and off the model surface, respectively. For the reconstruction of the mesh only the points k that provide zero responses are considered. In this way, coarse models can be reconstructed at different resolutions. This process is feasible if K is composed of points originating from the same distribution as the training samples p and q. Nevertheless, a raw PC containing only the coordinates of the vertices is insufficient to generate a 3D mesh. Thus, the marching cubes algorithm is employed for the 3D mesh generation from the predicted PC (Lewiner, Lopes, Vieira, & Tavares, 2003).

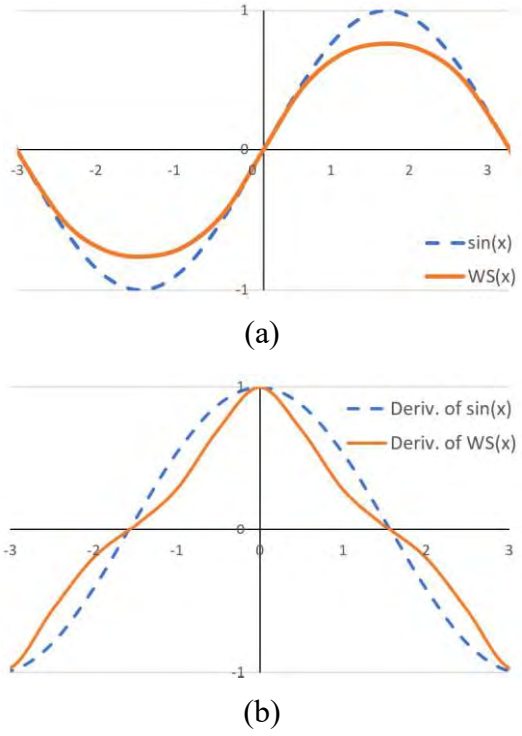


Figure 6.2. Graphical representation of the proposed WS and *sin* functions. (a) Responses and (b) derivatives of the *WS* and *sin* functions.

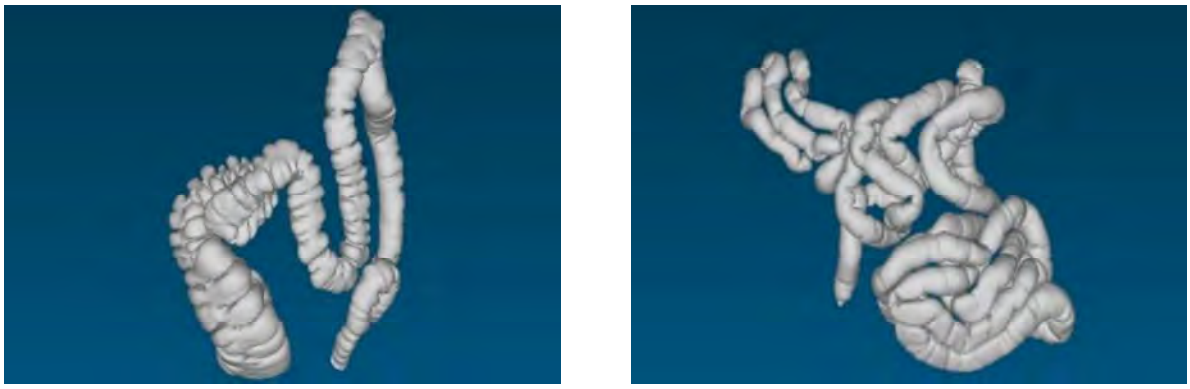
Recently, the utilization of the sinusoidal (*sin*) function for enhancing the performance of INRs has been proposed in (Sitzmann et al., 2020). The incorporation of the *sin* function to the neurons of an MLP has substantially improved its representation among different applications in the context of INR. The proposed approach introduces the WS function is aiming at further improving the implicit representation capacity of MLP regarding 3D models of the GI tract. The implementation of the proposed WS is achieved by applying the *tanh* function as a "wave-shaper" to the *sin* function. The *tanh* function is chosen since it is widely used as a wave-shaper in various applications in the context of signal processing (Huovilainen, 2004; Lazzarini & Timoney, 2010; Pakarinen & Yeh, 2009). The proposed WS activation function and its derivative are:

$$WS(x;\omega) = \tanh(\sin(\omega x)) \tag{6.1}$$

$$\frac{dWS(x;\omega)}{dx} = \omega \cos(\omega x)\, sech^2(\sin(\omega x)) \tag{6.2}$$

where $\omega \in \mathbb{R}$ is a learnable parameter of WS in contrast to (Sitzmann et al., 2020), where it is applied manually as a constant throughout the training and inference processes. Eq. (6.2) describes the first derivative of the WS function. As it can be seen in Figure 6.2 (a), for $\omega = 1$, the WS function can be regarded as a scaled approximation of *sin.* This means that WS maintains the properties of *sin*, *i.e.*, phase, periodicity, and it has upper and lower bounds. Nevertheless, it can be observed that the derivation of the WS function produces a more complex expression (Figure 6.2 (b)). Thus, it is expected that, during the training process, WS will produce a different computation of gradients compared to *sin*. In the following section, it will be demonstrated that the gradients computed during the training of an MLP that utilizes the WS function enable the network to efficiently reconstruct 3D models of the GI tract given only a small number of surface points.

Experiments and Results



(a) (b)

Figure 6.3. High-resolution 3D GI tract models. (a) Large intestine; (b) Small intestine.

To assess the reconstruction capacity of our methodology, a set of 5 different high-resolution 3D models of the large and small intestine was used (Clark et al., 2013). An example of these high-resolution models is illustrated in Figure 6.3. All model operations that were necessary for the evaluation process were performed in MeshLab v2021.10 (Cignoni et al., 2008). In detail, each model was represented by a dense PC that was automatically subsampled by Poisson disk sampling to generate sparse PCs of 0.5%, 1%, 5%, 10%, 20%, and 40% the original PC density. The models of each subsampling threshold, *e.g.*, the models comprising 0.5% of the points of the original high-resolution model, had the same ratio of points per unit area. The sub-sampling process was performed to simulate 3D models of the GI tract with a coarse 3D representation. This resulted in 30 different 3D PCs with various densities.

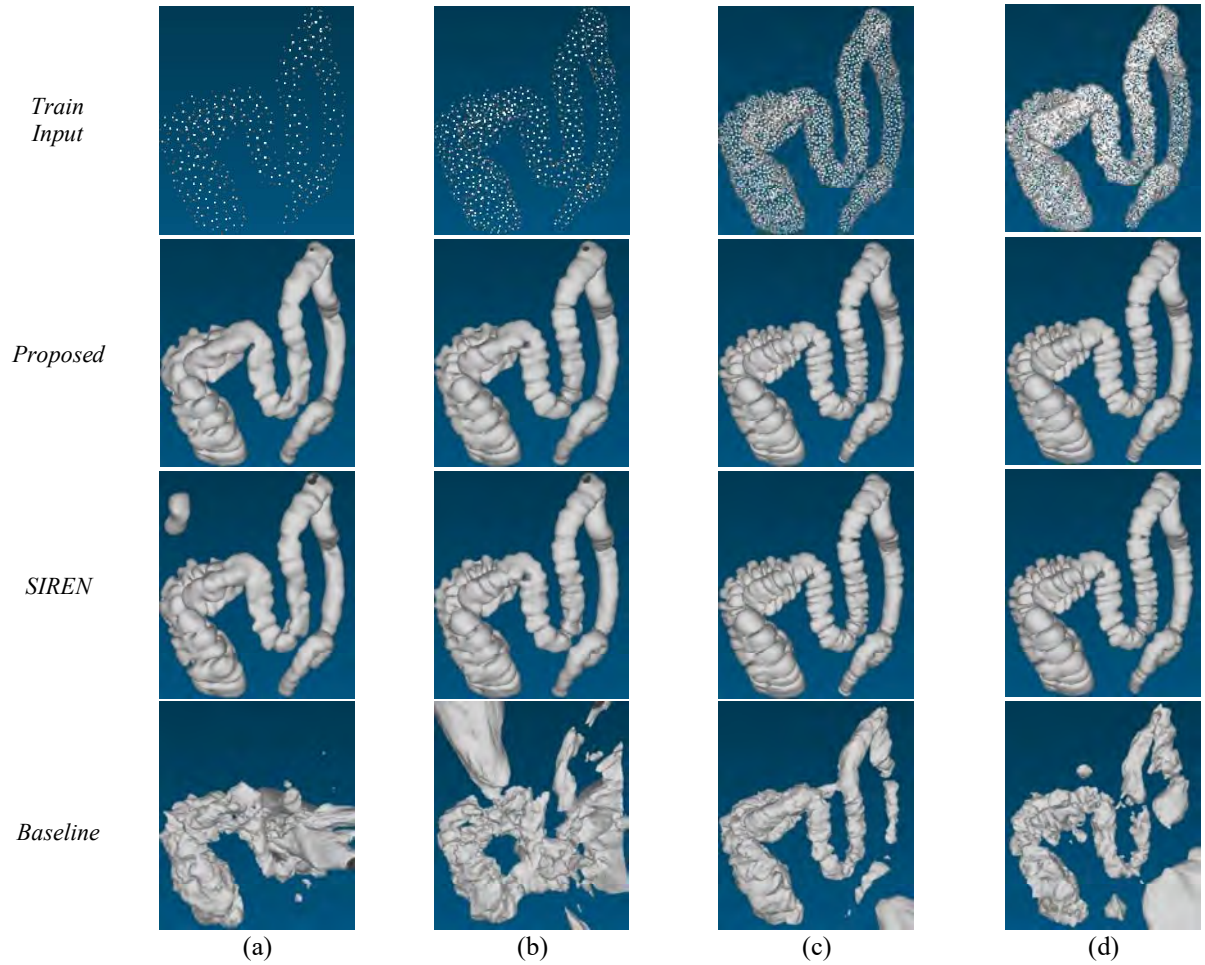
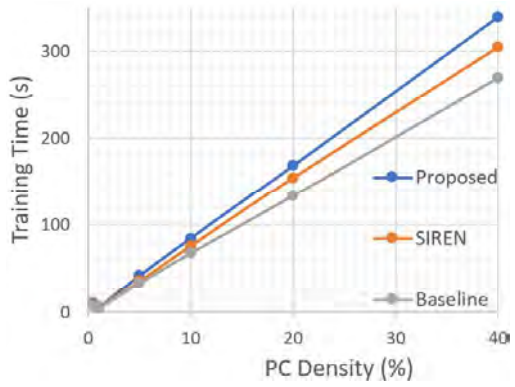


Figure 6.4. Qualitative comparison of the reconstruction outcome for a representative large intestine model reconstructed from PCs with various initial densities: (a) 0.5%; (b) 1%; (c) 5%; (d) 10%.

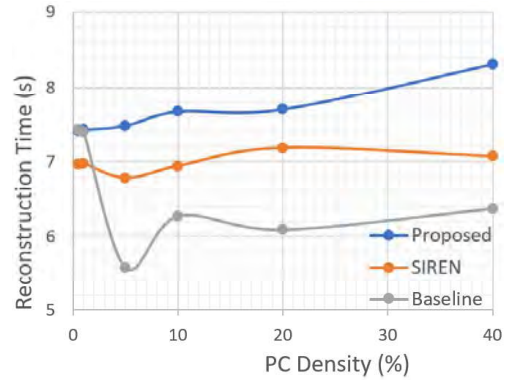
Table 6.1. Quantitative evaluation of the proposed WS activation function against other functions based on the CD and EMD metrics for different PC densities.

PC Density (%)	Methodologies					
	Proposed		*SIREN*		*Baseline (ReLU)*	
	CD	EMD	CD	EMD	CD	EMD
0.5	2.128	0.776	3.523	1.427	33.176	15.857
1	1.622	0.784	2.026	0.804	10.550	9.660
5	1.157	0.443	1.248	0.515	26.136	12.008
10	1.038	0.323	1.089	0.362	8.235	6.057
20	1.068	0.312	1.065	0.344	39.468	25.686
40	0.953	0.378	0.963	0.318	20.065	16.058

In all the experiments, the architecture of the neural network comprised 4 hidden layers, with 256 neurons each, utilizing the proposed WS activation function and a linear output layer. This type of MLP architecture has been widely used in related works (Ben-Shabat, Koneputugodage, & Gould, 2022; Sitzmann et al., 2020). All models that were used in the evaluation process were trained with



(a)



(b)

Figure 6.5. Graphs of the training and reconstruction time as a function of the PC density for the different methods. (a) Training time; (b) Reconstruction time.

the same number of epochs, batch size, and initialization parameters. The optimizer that was used to update the weights of the models was Adam, with an initial learning rate of 10^{-4}. Additionally, the reconstruction was performed at a resolution of 256^3. The experiments were conducted using a computer system equipped with an AMD Ryzen 5 3400G 3.70GHz processor, 16.00 GB RAM and the NVIDIA 2060 Super GPU.

The evaluation of the proposed method was performed with respect to the reconstruction accuracy of the high-resolution models given their coarse representations using the Chamfer Distance (CD) and Earth Mover's Distance (EMD) metrics (Levina & Bickel, 2001). Given two different PCs, CD is used to evaluate the distances between their closest points. EMD is used for the comparison of two different data distributions and is approximated by the 1^{st} Wasserstein Distance. Both CD and EMD are well-recognized metrics widely used to evaluate the performance of 3D reconstruction methods (Achlioptas, Diamanti, Mitliagkas, & Guibas, 2018; Z. Deng, Yao, Deng, & Zhang, 2021). These metrics are used to evaluate the similarity of two PCs in terms of the distance between their points. For comparison, SIREN (Sitzmann et al., 2020) was incorporated in the evaluation. In addition, an MLP with the aforementioned architecture, equipped with the Rectified Linear Unit (ReLU) (Agarap, 2018) was used as a baseline network in the comparative study.

Figure 6.4 presents a qualitative comparison among the results obtained by the different methodologies. As it can be observed, the use of the WS function improves substantially the refinement capacity of the MLP when the PC is highly sparse (Figure 6.4 (a, b, c)). However, as the PC of the model becomes denser (Figure 6.4 (d)), *i.e.*, more detailed, the results produced by the proposed methodology become comparable to (Sitzmann et al., 2020). In Figure 6.4 (a, b) it can be noticed that artifacts and distortions (*e.g.*, holes) are present in the model generated by SIREN. On the other hand, the proposed methodology utilizing the WS achieves a more detailed reconstruction with less noticeable distortions. Overall, the SIREN produces less detailed models compared to the WS-based method given highly-sparce PCs (Figure 6.4 (a, b)). Figure 6.4 (c, d) show that the baseline model can reconstruct the shape of the target 3D geometry, but the quality is worse compared to the other two methods. Table 6.1 summarizes the quantitative results

obtained from the comparative evaluation in terms of CD and EMD. The best results, which are obtained for lower CD and EMD values, are indicated in boldface typesetting. Overall, the quantitative results confirm the qualitative observations. It can be noticed that the proposed methodology results in a notably better performance with only 1% and 5% points of the original PC. As the PCs become denser, the proposed methodology produces comparable results with SIREN.

Figure 6.5 presents two graphs illustrating the training and reconstruction times required by each methodology to converge and generate meshes, respectively. Although the proposed methodology involves more complex computations (for the estimation of Eqs. (6.1) and (6.2)), the time it requires to be fully trained and reconstruct high-resolution meshes is of the order of the other approaches. Therefore, it has high-quality 3D reconstruction capabilities in a time-efficient manner.

6.2 Discussion and Future Work

In this section the use of INRs along with a novel periodic parametric activation function for the purpose of reconstructing coarse 3D models of the GI tract has been investigated. This approach examines the ability of a neural network to perceive and represent a data structure. After the training, a neural network is a continuous representation of that data structure and can be used to interpolate data points with high accuracy. To the best of our knowledge, this is the first time that INRs are exploited for the reconstruction of complex tissue models, such as the large and small intestine. The proposed method is based on a 5-layer MLP combined with a novel neural activation function. The effect of the proposed WS activation function was compared with state-of-the-art methodologies. The evaluation study suggests that the WS function can produce better results, both qualitatively and quantitatively, when evaluated in the context of 3D reconstruction of 3D models of the GI tract. More importantly, the WS function results in 3D models with significantly higher quality with only a few point observations. In addition, the parametrization of WS alleviates the need for its manual adjustment when the target model changes.

The proposed 3D reconstruction methodology can be regarded as unsupervised since it is trained directly on a PC of a coarse 3D model of the GI tract without the need for a huge dataset of other models and any labeled data. It can be used for the patient-specific reconstruction of various parts of the GI tract that can be incorporated in simulations or digital twins to assess critical clinical conditions, abnormality detection processes in the 3D domain, and pre-operative procedures.

Given the fact that periodic functions seem to have a positive effect on INRs, future work should focus on exploring other periodic activation functions and how different types of parameterizations affect the performance of the network. Moreover, the application of the WS activation function for the reconstruction of 3D models in different contexts or fields can be considered as a direction for future research.

Chapter 7
Perceptually Interpretable Convolutional Neural Networks

The discriminative power and remarkable performance of Convolutional Neural Networks (CNNs) on image classification comes along with a demand for consistent and meaningful interpretations that explain their decisions. The need for interpretation becomes more crucial with the application of CNN classifiers in high-stake domains, such as medicine (Diamantis, Iakovidis, & Koulaouzidis, 2019; Dimitris K Iakovidis, Georgakopoulos, et al., 2018) and autonomous driving (Kim & Canny, 2017; Y.-C. Liu et al., 2020). The interpretation of the inference process could earn the trust of the user by stripping away some of the opaqueness characterizing these black-box models (Castelvecchi, 2016).

A variety of different methods have been proposed to tackle the interpretation problem *post-hoc*. These include methods that construct proxy models with similar behavior to the original model aiming at extracting influential regions around a prediction (Ribeiro et al., 2016) or providing global behavioral explanations (Covert, Lundberg, & Lee, 2020). Another set of *post-hoc* techniques utilize the gradient that is backpropagated from the output prediction back to the input layer. These approaches construct saliency maps highlighting areas on the image that the network finds important (Qin et al., 2019; Selvaraju et al., 2016; Shrikumar, Greenside, & Kundaje, 2017; Smilkov, Thorat, Kim, Viégas, & Wattenberg, 2017; Sundararajan, Taly, & Yan, 2017; B. Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016). In general, *post-hoc* methods aim at interpreting the inference procedures of a deep learning algorithm after its development, while it has been documented that they could be unreliable in several cases (Rudin, 2019).

Approaching the problem from a different perspective, some methods aim towards the embedment of the interpretation mechanism into the design of deep learning models. Such methods are described as inherently interpretable and include decision trees, lists, sets, *etc.* (Lakkaraju et al.,

2016; Quinlan, 2014; R. L. Rivest, 1987). The structure of an inherently interpretable model is usually simpler; thus, its performance may be inferior to that of a more complex design that is interpreted *post hoc*. These approaches include feature disentanglement (Liang et al., 2020) and network dissection, which are aiming at quantifying the alignment of hidden variable-concept pairs (Bau et al., 2017) CNN models with feature guiding and self-attention mechanisms embedded in their architecture can also be regarded as inherently interpretable (Y.-C. Liu et al., 2020; Q. Zhang, Wu, & Zhu, 2018). These mechanisms enable the derivation of interpretations by visualizing saliency maps and CNN features that highlight important regions and concepts on the input image that guide a classification prediction (R. Chen et al., 2019).

Recently, inherently interpretable models that leverage the intelligibility and expressiveness of Generalized Additive Models (GAMs) have been proposed in the literature (Hastie & Tibshirani, 1990). These models use an architecture based on an ensemble of Multilayer Perceptrons (MLPs) (Agarwal, Frosst, Zhang, Caruana, & Hinton, 2020; Z. Yang, Zhang, & Sudjianto, 2020) to provide an interpretable outcome according to the contribution of each input feature to the ensemble consensus. CAM as a novel framework, extends the concept of the recently proposed MLP-based GAMs, for the interpretation of CNN-based image classification tasks of multi-class problems.

Considering the above, in the following section, a novel framework for constructing interpretable CNN models aiming at tackling computer vision tasks, named E Pluribus Unum Convolutional Neural Networks (EPU-CNN), is presented. EPU-CNN introduces a new direction towards interpretable CNN model design, leveraging the properties of additive models. The architecture of an EPU-CNN model consists of an ensemble of CNN subnetworks. Each sub-network receives a different representation of the input image, which expresses a perceptual feature of that image. A feature is considered perceptual if it can be easily perceived and interpreted by humans. For example, perceptual features include phenomenal representations, such as *hue* in HSV color space, and opponent representations, such as the component *a* (green-red color balance) of CIE-*Lab* (Wyszecki & Stiles, 2000). EPU-CNN predictions are the result of a discretized decision-making process performed by its CNN subnetworks. Other ensemble-based architectures are usually propagating the output of the subnetworks to fully connected modules, which obscures the contribution of each subnetwork to the final outcome (Y. Chen et al., 2019). On the contrary, the proposed aggregation is a linear process which, combined with the perceptual inputs, enables the direct interpretation of the classification procedure. Each subnetwork decides a classification outcome based on the perceptual feature that it receives as input. These decisions express the degree of similarity of an image to a particular class with respect to different perceptual features. EPU-CNN is easily scalable, since it consists of a number of sub-networks that can be adapted according to the needs of different applications.

7.1 Perceptually Interpretable CNN Model

The need for perceptual interpretation of image classification has motivated the development of a novel framework for the construction of inherently interpretable CNN models for computer vision tasks that is presented in this section. The proposed framework is called E Pluribus Unum Interpretable CNN (EPU-CNN), which means "out of many, one" interpretable CNN in Latin. The proposed framework has the significant advantage of being generic, in the sense that it can be used to make conventional CNN models interpretable. Following the GAM approach, an EPU-CNN model can be built as an ensemble of base CNN sub-networks given a base CNN architecture. According to EPU-CNN framework, each sub-network of the model must receive a set of complementary perceptual feature representations of the same input image. As a result, EPU-CNN is scalable since it can support an arbitrary number of parallel sub-networks corresponding to different perceptual features. The sub-networks are jointly trained and work together to generate interpretable class predictions automatically. An EPU-CNN model associates perceptual features with salient regions computed by the various sub-networks, and it explains a classification result by indicating the relative contribution of each feature to a predicted outcome.

To the best of our knowledge, EPU-CNN is the first GAM-based framework for building interpretable CNN ensembles, regardless of the base CNN architecture or application domain. Unlike current ensembles, EPU-CNN models enable interpretable classification based on both perceptual features and their spatial expression within an image; thus, the classification results can be interpreted more thoroughly and intuitively. It is worth noting that assembling shallower CNN architectures can be more efficient than training a single large model. (Kondratyuk, Tan, Brown, & Gong, 2020). Furthermore, unlike previous interpretable CNN models (Q. Zhang et al., 2018, 2019), the classification performance of EPU-CNN models is comparable to or higher than that of their non-interpretable counterpart, which in the case of EPU-CNN is the base CNN model. This is demonstrated with an extensive experimental evaluation on various biomedical datasets, including datasets from gastrointestinal endoscopy and dermatology, as well as a novel contributed benchmark dataset, inspired by relevant research in cognitive science (Deroy, 2013).

EPU-CNN framework follows the GAM approach for constructing image classification models capable of providing interpretations regarding their results. GAMs extend the well-known linear regression models by incorporating to their architecture a sum of smooth functions $\sum f_i(x_i)$, i = 1, 2, …, N. A GAM can be formally expressed as:

$$g(\mathbb{E}[Y \mid x]) = \beta + \sum_{i=1}^{N} f_i(x_i) \qquad (5.1)$$

where $x = (x_1, x_2, \ldots, x_N)^T$, $x \in \mathbb{R}^N$, denotes an input feature vector, $g(\cdot)$ is a link function (*e.g.*, logit), β is a bias term and $\mathbb{E}[Y \mid x]$ denotes the expected value of the response variable Y, given an input x. Each $f_i(\cdot)$, represents a univariate smooth function, $f_i: \mathbb{R} \rightarrow \mathbb{R}$, mapping each $x_i \in \mathbb{R}$ to $f_i(x_i)$, that reflects x_i participation to the predicted output of the model. This structure provides a simple

solution to interpretability of the model since it allows the user to investigate how each input variable x_i affects the predicted output.

For the construction of its interpretable ensemble of CNNs, EPU-CNN framework considers Eq. (5.1) as a template (Figure 7.1). The arrangement of the sub-networks is parallel, and all the sub-networks have identical architectures. The architecture that is used by the sub-networks is referred to as the base model. Each input that is propagated to the different sub-networks should encode a perceptual feature representation of an image. This representation is referred to as the Perceptual Feature Map (PFM) of an input image, and it can be obtained by performing an image transformation that reveals a physical property of choice that can be easily perceived and interpreted by humans over the input image space, such as color and texture. (Greisdorf & O'Connor, 2002). The number of sub-networks is based on the number of different PFMs needed to make a CNN interpretable for a specific application. Given that each sub-network of an ensemble with a parallel topology should receive inputs with complementary information (Woniak, Grana, & Corchado, 2014), the PFMs should be orthogonal. Let N different PFMs I_i, $i = 1, 2, \ldots, N$, of an input image I be the input that is propagated to a respective sub-network $C_i(\cdot\,; \eta_i)$, where η_i denoting its parameterization. All the sub-networks are trained jointly as a single model; hence, the input of a model constructed according to EPU-CNN framework is a tensor $I = (I_1, I_2, \ldots, I_N)$ with dimensions of $N \times H \times W$, where N, H, and W denote the number, height, and width of the PFMs I_i, respectively. Each sub-network provides a univariate output $C_i(I_i; \eta_i)$. The output of the EPU-CNN ensemble is computed as the sum of all $C_i(I_i; \eta_i)$, $i = 1, 2, \ldots, N$. Each output $C_i(I_i; \eta_i)$ is considered as a Relative Similarity Score (RSS) that quantifies the relative similarity of an image I to a class with respect to I_i. In a classification problem that incorporates n classes, RSS receives n values that fall within the range [-1, 1] for each class. An absolute RSS value closer to 1 indicates greater similarity, whereas a positive or negative RSS sign indicates that

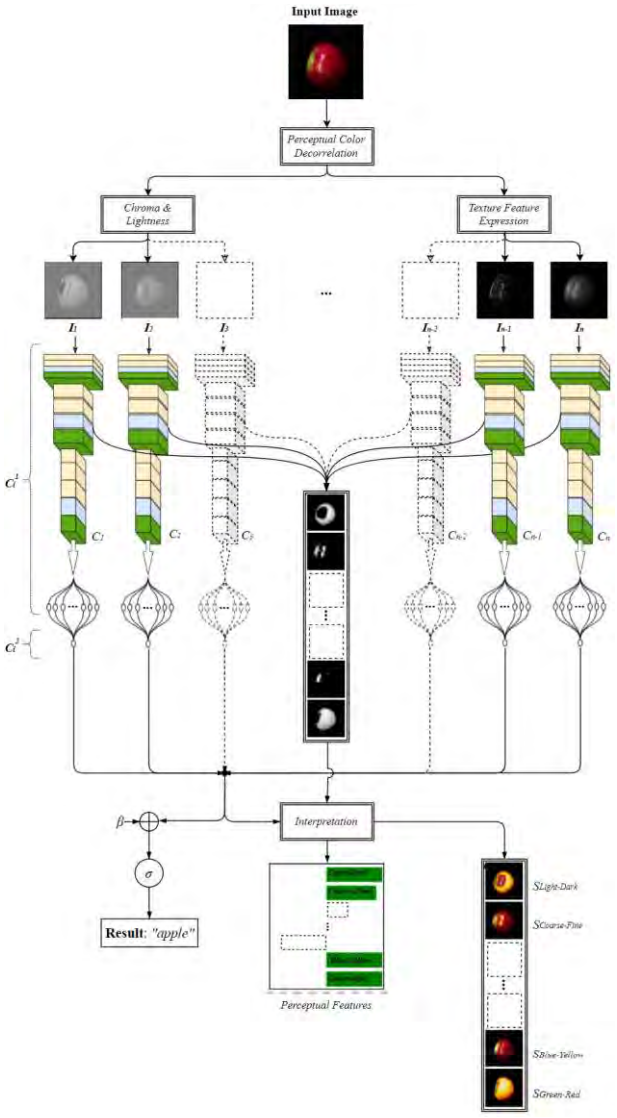


Figure 7.1. Outline of the EPU-CNN framework.

the similarity is associated with one class or the other. With the visualization of RSS scores, a human can better understand how each I_i influences the EPU-CNN model's classification result. Furthermore, the scores can be associated with respective image regions by examining the layer activations of each subnetwork, allowing for a more in-depth interpretation of the classification result based on the spatial arrangement of the observed features within the input image. The following paragraphs describe the PFMs considered in this study, the formulation of the classification model, and its interpretable output.

Opponent Perceptual Feature Maps

The generation of PFMs in this study is motivated by Hering's theory of human perception of color vision proposed in the 1800s and Hurvich and Jameson's opponent-process theory proposed in the 1950s. (Hurvich & Jameson, 1957). A receptive field is a pattern of photoreceptors that determines the behavior of a cell in the human visual system's retina. A cell that is excited by a light stimulus in the center of its receptive field will be inhibited by a light stimulus in the annulus surrounding the excitatory center since receptive fields have a center-surround organization. Photoreceptors are classified according to their sensitivity to light frequency and intensity, as well as their response to chromatic and luminance variations. Receptive fields can be color-opponent or spatially-opponent without being color-opponent, depending on the type of photoreceptors (Chatterjee & Callaway, 2003). Studies have shown that transmitted stimuli to the retina can be decomposed into independent luminance and chromatic-opponent sources of information, and that the human visual system processes chromatic and luminance information separately (Mäenpää & Pietikäinen, 2004; Poirson & Wandell, 1996). Furthermore, experiments that have been conducted in the context of computer vision shown that encoding the chromatic and luminance components separately is beneficial for image recognition tasks (Hansen & Gegenfurtner, 2009; Mäenpää & Pietikäinen, 2004).

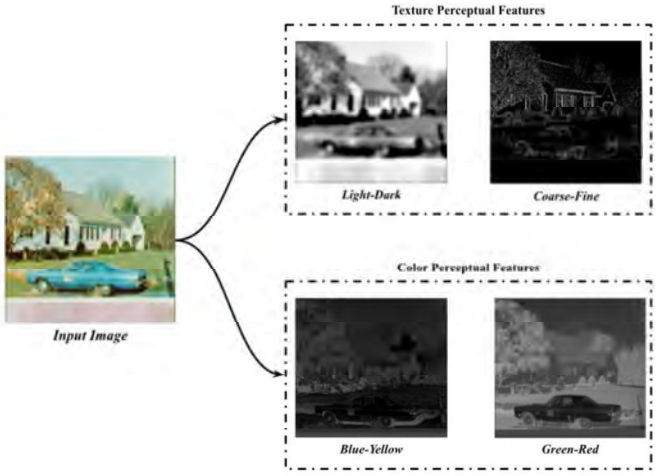


Figure 7.2. Illustration of the opponent perceptual features utilized by EPU-CNN.

Motivated by these studies, the proposed framework takes into account representations of the input images that are characterized as opponent, focusing on two important properties for image understanding, i.e., color and texture (Greisdorf & O'Connor, 2002). Color and texture also provide cues that allow inferences about the shapes of objects and surfaces in the image. Opponent color spaces have been proposed to address the shortcomings of RGB color space that include high correlation between the R, G, and B components and its incompatibility with human perception.

Ohta's color space, which is obtained as a linear transformation of RGB and has been proposed in the context of color image segmentation, and CIE-Lab, which is obtained as a non-linear transformation of RGB and has been proposed as a device independent, perceptually uniform color space, are two representative examples (i.e., a color space where a given numerical change corresponds to similar perceived change in color) (Wyszecki & Stiles, 2000). Motivated by the effectiveness of CIE-*Lab* in many computer vision tasks and especially in biomedicine (Dimitris K Iakovidis & Koulaouzidis, 2014), this study considers CIE-*Lab* as a basis for three PFMs that correspond to tis components. Moreover, the orthogonality that characterizes the components of CIE-*Lab* is suitable for the generation of PFMs in the context of EPU-CNN framework. The *a* and *b* components of CIE-*Lab* encode two antagonistic colors that cannot be perceived together simultaneously, *e.g.*, there is no "reddish-green" or "bluish-yellow" color. In detail, *a*, expresses the antagonism between *green-red* hues (redness is expressed for $a > 0$, and greenness is expressed for $a < 0$), and component *b* expresses the antagonism between *blue-yellow* hues (yellowness is expressed for $b > 0$, and blueness is expressed for $b < 0$). The L component represents perceptual lightness, which expresses a luminance antagonism between light and dark. This component, which is essentially a greyscale representation of the RGB image, has the highest variance because it concentrates information regarding the texture of the contents of an image (Mäenpää & Pietikäinen, 2004).

In the computer vision field, many studies have been based on the representation of the spatial frequency of images, for the modeling of texture and its utilization in machine perception applications (P.-W. Huang & Dai, 2003). Towards the interpretation of the classification outcomes based on characteristics of perceptual texture, the lightness component *L* of CIE-*Lab* is further decomposed with respect to its spatial frequency. This decomposition is justified by the capacity of the human eye to focus on the appropriate range of spatial frequencies with the goal of capturing relevant details that describe an image; hence, visual perception treats images at different levels of resolution. Details at lower resolutions correspond to larger structural elements of a scene, whereas at details at higher resolutions encode smaller structures. The 2D Discrete Wavelet Transform can model the concept of multiresolution image representation (DWT) (Mallat, 2009). The original image is decomposed using a wavelet orthonormal basis to yield this representation. The à trous algorithm, which is based on convolutions of the image with a pair of low and high-pass filters called Quadrature Mirror Filters (QMFs) while along with dyadic down-sampling, is used to compute the 2D DWT efficiently. A multilevel 2D DWT can be performed by successively applying the 1-level 2D DWT to the filtered image with the lowest frequencies and focusing on different bands of non-overlapping spatial frequencies. The last level's lowest frequency image represents a smooth approximation of the input image, where the different structures, such as objects, parts of objects, and background, can be more easily separated based on their intensity. Based on this observation, the approximation image of the third level of the 2D DWT was chosen as a PFM representing the light-dark antagonism with less noise than the original L component in the context of EPU-CNN. The higher frequency bands can be used as PFMs representing image texture in greater detail. The frequency bands chosen are determined by the application context

and the level of the desired interpretation detail. The highest frequency band of the first level of the 2D-DWT was chosen in this study to represent the antagonistic concept of coarse-fine texture due to the fact that the edges of an image become clearer at that level. The density of image edges per unit area is associated with the concept of coarse-fine texture (Tuceryan & Jain, 1993), because finer textures have a higher edge density per unit area than coarser textures. Such edge-based representations are perceptually meaningful for object discrimination in a scene (Biederman & Ju, 1988). Given that after each level of the 2D DWT the resolution of the image of the previous level is reduced by a factor of 2, and that the base CNN model architecture depends on the dimensions of the input image, the filtered images obtained after the application of the 2D DWT are scaled to match the size of the input image *I*. Henceforth, an input tensor of an EPU-CNN model is formed as $I = (I_1, I_2, I_3, I_4)$, where I_1 and I_2 are the PFMs corresponding to the *light-dark* and *coarse-fine* concepts respectively, $I_3 = b$ corresponds to *blue-yellow*, and $I_4 = a$ corresponds to the *green-red* concept. An example illustrating the opponent PFMs used in this study, is provided in Figure 7.2.

Binary Interpretable Classification Model

Given an input tensor *I* composed of *N* input PFMs, an EPU-CNN model performs feature extraction and classification. An EPU-CNN model is constructed from *N* CNN sub-networks C_i, with each sub-network receiving a PFM $I_i, i = 1, 2, \ldots, N$, as input. Each C_i can be regarded as a function $C_i(\cdot\,; \eta_i)$, $C_i : X^{H\times W} \rightarrow Z$, where $X^{H\times W}$ and Z are the input and univariate output space of each C_i, respectively. A sub-network C_i consists of two parts: *a)* a feature extractor $C_i^1(\cdot\,; \omega_i)$ parametrized by ω_i; and *b)* a univariate function $C_i^2(\cdot\,; \theta_i)$ parametrized by θ_i. Thus Eq. (5.1) is reformulated:

$$g(\mathbb{E}[Y \mid I; \theta_{\{N\}}, \omega_{\{N\}}]) = \beta + \sum_{i=1}^{N} C_i^2(C_i^1(I_i; \omega_i); \theta_i) \tag{5.2}$$

where C_i^1 represents a feature extraction model composed of a CNN followed by a Fully Connected Neural Network (FC-NN), that utilizes activation functions, which are not conditioned to be smooth, and C_i^2 represents a single FC-NN layer utilizing a smooth activation function that provides the final univariate output of a CNN sub-network, $\omega_{\{N\}} = \{\omega_1, \omega_2, \ldots \omega_N\}$ and $\theta_{\{N\}} = \{\theta_1, \theta_2, \ldots \theta_N\}$ are the parameters of C_i^1, C_i^2, respectively. Equation (5.2) encapsulates the properties and definition of GAMs while extending its capacity to exploit CNN models for computer vision tasks. The feature extractor C_i^1, can be implemented by a conventional CNN architecture, whereas the number of output neurons and the activation function of C_i^2 should be considered so to appropriately represent the classification outcome, in a binary or multiclass setting. In the context of binary classification, which is considered in this study, C_i^2 is formulated with a single output neuron and the hyperbolic tangent (*tanh*) activation function, resulting in sub-network responses within the range of [-1, 1]. Ultimately, this allows to intuitively express the contribution of each feature to the final prediction, as positive or negative contribution with respect to a class label. Since EPU-CNN is applied in the context of binary classification, we chose the final output of an

EPU-CNN model to be within the interval of [0, 1]. However, the formulation of EPU-CNN presented in Eq. (5.2) indicates that the final output can fall out of the range of [0, 1], *i.e.*, given a N number of C_i and a bias term β the right-hand part of Eq. (5.2) provides values that fall within the range of $[\beta - N,\ \beta + N]$.

Accordingly, the *logit*(·) function, defined as:

$$logit(x) = -log\left(\frac{1}{x} - 1\right) \tag{5.3}$$

can be used as a suitable link function, $g(\cdot)$. Considering that the inverse of *logit* is the *log-sigmoid* function σ, Eq. (2) can be rewritten as:

$$\mathbb{E}[Y \mid I; \theta_{\{N\}}, \omega_{\{N\}}] = logit^{-1}\left(\beta + \sum_{i=1}^{N} C_i^2(C_i^1(I_i; \omega_i); \theta_i)\right) \tag{5.4}$$

or

$$EPU_{CNN}(I;\ \eta_{\{N\}}) = \sigma\left(\beta + \sum_{i=1}^{N} C_i(I_i; \eta_i)\right) \tag{5.5}$$

where $\eta_{\{N\}} = \{\eta_1, \eta_2, \ldots, \eta_N\}$. By utilizing the *log-sigmoid* function we bound the output of EPU-CNN within the desirable range of [0, 1] suitable for binary classification applications. Equation (5.5) is a formal representation of an EPU-CNN model as illustrated in Figure 7.1. To train an EPU-CNN model in the context of binary classification, the Binary Cross Entropy (BCE) is chosen as a loss function to be minimized:

$$EPU_{CNN}: \begin{cases} argmin_{\eta_1}\left(-\frac{1}{k}\sum_{j=1}^{k} y_j log\left(EPU_{CNN}(I_j;\ \eta_{\{N\}})\right) + (1-y_j) log\left(1 - EPU_{CNN}(I_j;\ \eta_{\{N\}})\right)\right) \\ \vdots \\ argmin_{\eta_N}\left(-\frac{1}{k}\sum_{j=1}^{k} y_j log\left(EPU_{CNN}(I_j;\ \eta_{\{N\}})\right) + (1-y_j) log\left(1 - EPU_{CNN}(I_j;\ \eta_{\{N\}})\right)\right) \end{cases} \tag{5.6}$$

where $j = 1, 2, 3\ldots, k$, $EPU_{CNN}(I_j; \eta_{\{N\}})$ is the class probability of I_j and y_j is the ground truth label of I_j. As it can be observed from Eq.(5.6), the total error of the EPU-CNN, deriving from the responses of the CNN ensemble consensus, is used to update the parameters of each $C_i(\cdot; \eta_i) = C_i^2(C_i^1(\cdot; \omega_i); \theta_i)$ of the parallel sub-network ensemble topology, simultaneously. It is worth noting that an EPU-CNN model can also be adapted for multiclass datasets, *e.g.*, using n>1 output neurons instead of one, in the case of $n > 1$ classes. Then, the network's output can be interpreted by considering the contribution of the multiclass classification outcome of each CNN sub-network to the final classification result (see Multiclass Interpretable Classification Model).

Multiclass Interpretable Classification Model

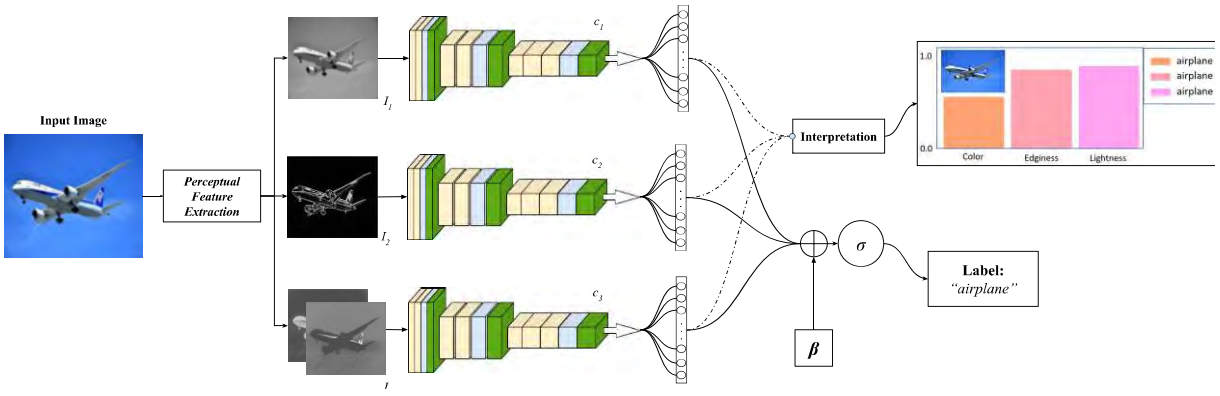


Figure 7.3. Illustration of multiclass version of the architecture of an EPU-CNN model

EPU-CNN can also be used for the development of interpretable CNNs that can be used in solving problems that incorporate data that belong to multiple classes, *i.e.*, multi class problems. These EPU-CNN models utilize subnetworks that can be expressed as functions $\Phi_i : X^{H\times W\times D} \rightarrow Y^M$, where H, W and D denote the dimensions of height, width, and depth of the input volume, respectively, whereas $X^{H\times W\times D}$ and Y^M denote the input and output space, respectively. The output space of each subnetwork is M dimensional, where M is the number of classes. The input of these EPU-CNN models is the same as to those that are tasked to tackle binary classification problems, *i.e.*, a tensor $I = (I_1, I_2, I_3, \ldots, I_N)$ with dimensions of $N\times H\times W\times D$. Hence, the EPU-CNN models for multiclass problems has the following general form that is similar to Eq. (5.5):

$$EPU_{multi}(I;\eta_{\{N\}}) = softmax\left(b + \sum_{i=1}^{N} \Phi_i(I_i;\eta_i)\right) \tag{5.7}$$

By applying the *softmax* activation function to the right part of Eq. 2, the output is bounded within the desirable interval. Equation (5.8) is a formal representation of an EPU-CNN model designed for multiclass problems as illustrated in Figure 7.3. To train of an EPU_{multi} model, a joint loss function is used that considers the classification error of both EPU_{multi} as a whole and its individual subnetworks. In detail, to assess the classification error of a model constructed according to the proposed framework, the Categorical Cross-Entropy (CCE) is applied during the training on both EPU_{multi} and its subnetworks:

$$EPU_{multi}: argmin_{\eta_{\{N\}}}\left(CCE(EPU_{multi}(I;\eta_n), y) + \sum_{i=1}^{N} CCE(\Phi_i(I_i;\eta_i), y)\right) \tag{5.8}$$

where EPU_{multi} $(I; \eta_{\{N\}})$ and $\Phi_i(I_i)$ are the class probabilities of I as estimated by a EPU_{multi} model and the decisions of its subnetwork Φ_i given an image representation F_i, respectively. The term y is the categorical representation of the ground truth label of I. According to Eq. (5.8), the ensemble is trained jointly through back-propagation and each subnetwork learns to extract features that correspond to a particular perceptual domain of the input image. In this way, each subnetwork attempts to classify an input image with respect to its corresponding domain of knowledge. Hence, the output of each subnetwork of EPU_{multi} provides a classification decision with respect to the image representation that it receives as input. The aggregation of these decisions comprises the final classification result of EPU_{multi}.

Perceptual Interpretable Output

Considering an input image, EPU-CNN provides three outputs, as illustrated in Figure 7.1 and Figure 7.2, namely, *a)* the predicted class $EPU_{CNN}(I; \eta_{\{N\}})$; *b)* a set of RSSs $C_i(I_i; \eta_i)$, $i = 1, 2, \ldots, N$, explaining why the image is classified in that class; and *c)* a set of Perceptual Relevance Maps (*PRMs*) S_i explaining which image regions are responsible for each RSS. Figure 7.4 illustrates the provided outputs of the model for two images that belong to different classes. The classification result is indicated as a textual label characterizing the input image, and the RSSs are visualized through bar-charts. Each bar-chart consists of horizontal red or green colored bars, indicating the magnitude of resemblance that each I_i is estimated to have for the banana and apple class, respectively. Additionally, the model provides with respect to each I_i, areas (PRMs) highlighting their resemblance to the predicted class. The color scaling from orange to yellow regions of the maps indicates the ascending intensity of activation.

Image-specific visualizations of RSSs enable the interpretation of the classification process of unlabeled input images. This is the most important aspect of an EPU-CNN model. For example, the image of Figure 7.4 (a), is classified as a banana, because all PFMs, *i.e., light-dark*, *coarse-fine*, *blue-yellow* and *green-red*, as

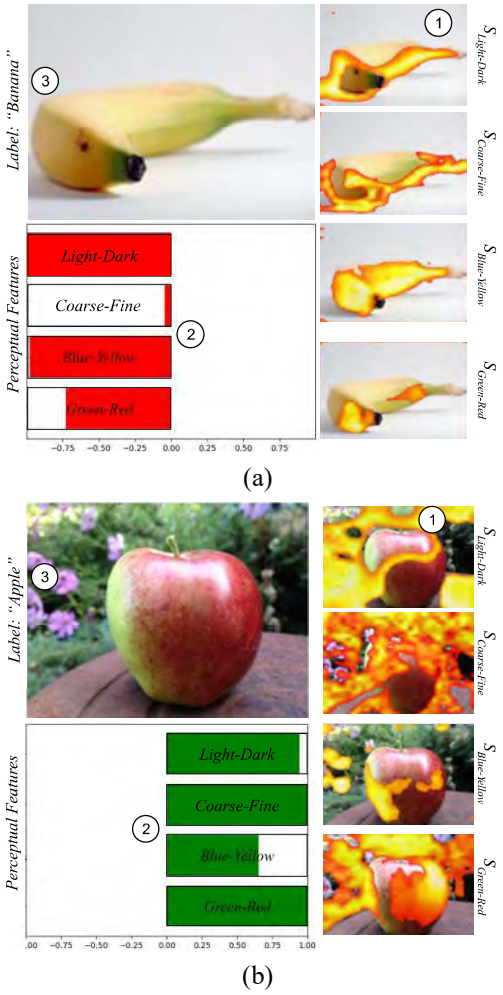


Figure 7.4. Example of EPU-Net output visualization using bar-charts and saliency maps. The numbering indicates the interpretation order of EPU-CNN output. The label field indicates the

indicated by the respective RSSs, guide the prediction towards the banana class, which corresponds to negative $C_i(I_i; \eta_i)$ responses (red). Accordingly, the image of Figure 7.4 (b), is classified as an apple, because all PFMs guide the prediction towards the apple class, *i.e.*, positive $C_i(I_i; \eta_i)$ responses (green). However, it is not necessary for all the $C_i(I_i; \eta_i)$ responses to be negative for an image to be classified as a banana, since EPU-CNN models consider the consensus of the sub-networks.

Perceptual Relevance Maps, S_i, are generated to visually inspect the relevant regions of the input image I with respect to each RSS $C_i(I_i; \eta_i)$. Let $F_i^l = (f_{i,l}^1, f_{i,l}^2, f_{i,l}^3 \dots, f_{i,l}^n)$ indicate a tensor of feature maps with $F_i^l \in \mathbb{R}^{n \times h \times w}$, where n, h, w denote the depth, height and width of F_i^l, and $f_{i,l}^n \in \mathbb{R}^{h \times w}$, as computed by a convolutional layer l of a C_i. The selection of l is intertwined with its capacity to highlight regions that contribute to the derivation of $C_i(I_i; \eta_i)$. The deeper the layer l that F_i^l is extracted from, the more approximate the correspondence among the feature maps $f_{i,l}^n$ and the input image I; thus, a middle layer l of C_i is considered for the construction of each S_i (Dimitris K Iakovidis, Georgakopoulos, et al., 2018) (Section 3.4).

To quantify the amount of information that each $f_{i,l}^n$ encodes, we compute the Shannon Entropy (SE) scores. Then, half of the most informative $f_{i,l}^n$, *i.e.*, $f_{i,l}^n$ that correspond to the highest entropy scores, are aggregated to construct the S_i. The aggregation is performed by averaging $f_{i,l}^n$ features maps which results to the initial S_i estimation. Then S_i is further refined, by applying a thresholding method that maximizes the entropic correlation between the foreground and background of S_i, for maximum information transfer (Yen, Chang, & Chang, 1995). The entropy-based thresholding operation is performed to exclude values associated with lower saliency and communicate to the user the most informative regions. An example of different S_i of input images I can be seen in Figure 7.4. The generated S_i illustrated in Figure 7.4 are overlayed on the input images (Figure 7.4 (a, b)). The highlighted regions indicate the spatial association of similarity scores $C_i(I_i; \eta_i)$ with the respective input image. Moreover, the numbers in the images of Figure 7.4 indicate in which order the different outputs of an EPU-CNN can be considered by the user. Initially a user can examine the regions that are highlighted by the generated PRMs of each PFM (1). Subsequently, these regions are participating to the classification outcome, towards either class, with a magnitude that is indicated by the RSSs (2). Finally, the PRMs (1) along with the RSSs (2) can assist the user to interpret the class prediction of an EPU-CNN (3).

Experiments and Results

Datasets: The training and evaluation of EPU-CNN incorporates six different datasets. Initially, a dataset created specifically for assessing the interpretability capabilities of EPU-CNN was considered. The goal of using this dataset was to demonstrate EPU-CNN capabilities using clear, simple, and perceptually meaningful examples. By taking into consideration the importance of biomedicine as a critical application area for explainable and interpretable artificial intelligence (AI), four well-known biomedical benchmark datasets comprising endoscopic and dermoscopic

images were used for further evaluation. Lastly, a well-known benchmark dataset for real image classification was also considered to demonstrate the generality of the proposed approach.

Interpretability Dataset: A new dataset called Banapple was created for the purposes of this study consisting of images illustrating bananas and apples . by collecting images from Flickr that were licensed under the Creative Commons license. The images depict bananas and apples with varying color, placement, size, and background. The inspiration for this dataset comes from cognitive science studies in which human perception is investigated using examples with discrete properties of bananas and apples (Deroy, 2013). The performed experiments aim at demonstrating that EPU-CNN is capable of capturing the discriminative characteristics of bananas and apples by the perceptual features it incorporates, *i.e.*, apples have a circular shape and usually red color, whereas bananas have a bow-like shape and usually a yellow color. In addition, samples that deviate from the average appearance of these objects can provide insights regarding the reliability of the interpretation of the model.

Endoscopic Datasets: This evaluation includes publicly available datasets of endoscopic images. Namely, KID (Anastasios Koulaouzidis et al., 2017), Kvasir (Pogorelov et al., 2017) and a dataset that was part of the MICCAI 2015 Endovis challenge (Navab, Hornegger, Wells, & Frangi, 2015). KID dataset comprises 2,352 annotated wireless capsule endoscopy (WCE) images of abnormal findings, *i.e.,* inflammatory, vascular and polypoid lesions as well as images depicting normal tissue from the esophagus, stomach, small bowel and colon. The Kvasir dataset consists of images of the gastrointestinal (GI) tract, annotated and verified by medical experts. These include 4,000 images of anatomical landmarks, *i.e.,* Z-line, pylorus and cecum, and pathological findings of esophagitis, polyps and ulcerative colitis. The dataset also contains sets of images related to endoscopic polyp removal that were not utilized for this work. The MICCAI 2015 Endovis challenge dataset consists of 800 gastroscopic images of normal and abnormal findings, such as gastritis, ulcer, and bleeding.

Dermoscopic Dataset: The evaluation process of EPU-CNN has also included the International Skin Image Collaboration Challenge 2019 (ISIC2019) dermoscopic image collection. ISIC2019 challenge provides a publicly available archive of 25,331 dermoscopic images of eight different categories of skin lesions, namely, melanoma, melanocytic nevus, carcinomas (both of basal and squamous cells), actinic and benign keratosis, dermatofibroma, and vascular lesions. These images were used to construct three different binary classification problems: *a)* melanomas *vs.* melanocytic nevi (*Me. vs. Ne.*); *b)* carcinomas *vs.* melanocytic nevi (*Ca. vs. Ne.*) and *c)* carcinomas *vs.* melanomas (*Ca. vs. Me.*). The tasks *a)* and *b)* are characterized as a classification between abnormal (carcinomas, melanomas) and normal (melanocytic nevus) skin lesions whereas task *c)* discriminates two abnormal categories of different incidence and survival rates, *i.e.*, melanomas have higher mortality rates than carcinomas (Siegel, Miller, Fuchs, & Jemal, 2021). Task *a)* comprised of 9000 images whereas task *b)* and *c)* 8200 and 8500 images respectively.

CIFAR-10: To demonstrate the generality of the proposed framework, EPU-CNN was further validated on the long-standing benchmark dataset CIFAR-10. CIFAR-10 consists of 60,000 color

images of natural objects that belong to 10 different classes. The dataset is split in 50,000 training and 10,000 test images and each class comprises 6,000 images with a size of 32 × 32.

Classification Performance Assessment

For the comparison of the classification performance of EPU-CNN, we selected three well established CNN models, namely, VGG16 (Simonyan & Zisserman, 2014), ResNet50 (He et al., 2016) and DenseNet169 (G. Huang et al., 2017) and an inherently interpretable CNN model abbreviated as TT (C. Chen et al., 2019). VGG16 was used as a base for the TT model. The same training parameters, *i.e.*, batch size, optimization algorithm and data augmentation, were applied on all networks involved in the evaluation process. In detail, the batch size was set to 64 and as an optimization algorithm the Stochastic Gradient Decent was used; the training data were augmented only with respect to their orientation. The weights of all networks were randomly initialized before training. Five different CNNs architectures were considered for the construction of EPU-CNN models. In detail, two indicative CNN architectures, namely, *BaseI* and *BaseII*, along with *VGG16, ResNet50* and *DenseNet169* were incorporated as base models in the EPU-CNN framework.

These models were selected to demonstrate the generality of the proposed framework, i.e., its applicability to rendering different conventional CNN architectures interpretable. Regarding the architecture of the indicative CNN architectures, $Base_I$, consists of 3 convolutional blocks in total, followed by an FCNN. The first two convolutional blocks are identical and include two convolutional layers followed by a max-pooling and a batch normalization layer. The convolutional layers of these blocks have a depth size of 64 and 128 respectively. The following convolutional block consists of three convolutional layers with a depth size of 256 followed by a max-pooling and a batch normalization. All the kernels of the convolutional layers had a size of 3×3. $Base_{II}$, follows the same architecture with $Base_I$ with an additional convolutional block, in the beginning of the architecture, utilizing an inception module. $Base_I$, $Base_{II}$, VGG_{16}, $ResNet_{50}$ and $DenseNet_{169}$ were used for the construction of EPU_I, EPU_{II}, EPU_{VGG}, EPU_{ResNet} *and* $EPU_{DenseNet}$, respectively.

The evaluation followed a 10-fold cross validation procedure with the average Area Under the receiver operating Characteristic (AUC) score among all folds. The AUC was selected as an overall summary measure of binary classification performance, which unlike accuracy, is relatively robust for datasets with imbalanced class distributions (Provost & Fawcett, 1997). The performance of all models is summarized in Table 7.1. The best results are in boldface typesetting and the results ranked second are underlined. It can be observed that the results obtained by the EPU-CNN models indicate an overall better or comparable classification performance to their non-interpretable counterparts, *i.e.*, $Base_I$, $Base_{II}$, VGG_{16}, $ResNet_{50}$ and $DenseNet_{169}$. In detail, on Banapple, Endovis-MICCAI, Kvasir and ISIC 2019 (*Me. vs. Ne.*) EPU_{II} provided substantially better results when compared to the other EPU-CNN and the majority of base models. Figure 4 illustrates the number of trainable parameters of each model. It can be observed that the complexity of an EPU-CNN model is analogous to that of its base model. Additionally, an EPU-CNN can

Table 7.1. Classification Results (AUC) of EPU-CNN and CNN models

Models	Datasets						
	Banapple	*KID*	*Endovis*	*Kvasir*	ISIC 2019		
					Ca.vs.Ne.	*Ca.vs.Me.*	*Me.vs.Ne.*
EPU_{I}	0.91±0.01	0.94±0.02	0.97±0.01	0.87±0.02	0.96±0.03	0.92±0.01	0.94±0.02
EPU_{II}	**0.92±0.01**	0.93±0.01	**0.97±0.01**	**0.92±0.01**	0.94±0.02	0.91±0.01	**0.94±0.01**
EPU_{VGG}	0.90±0.01	0.93±0.01	0.89±0.03	0.88±0.01	**0.97±0.04**	0.90±0.01	0.89±0.03
EPU_{ResNet}	0.84±0.04	0.86±0.06	0.84±0.01	0.79±0.04	0.88±0.08	0.78±0.06	0.86±0.04
$EPU_{DenseNet}$	0.90±0.03	0.93±0.05	0.87±0.01	0.90±0.03	0.97±0.03	0.92±0.02	0.92±0.03
$Base_{I}$	0.92±0.02	0.96±0.02	0.96±0.01	0.91±0.01	0.90±0.02	**0.94±0.03**	0.93±0.02
$Base_{II}$	0.91±0.01	**0.97±0.01**	0.97±0.01	0.91±0.01	0.93±0.01	0.92±0.04	0.93±0.04
VGG_{16}	0.90±0.01	0.90±0.04	0.93±0.01	0.85±0.01	0.87±0.01	0.90±0.03	0.92±0.02
$ResNet_{50}$	0.89±0.02	0.92±0.03	0.88±0.10	0.87±0.09	0.69±0.12	0.90±0.02	0.92±0.03
$DenseNet_{169}$	0.88±0.04	0.94±0.05	0.90±0.12	0.88±0.01	0.76±0.09	0.90±0.01	0.91±0.03
TT_{VGG}	0.82±0.04	0.91±0.03	0.93±0.05	0.85±0.04	0.88±0.03	0.76±0.01	0.81±0.02

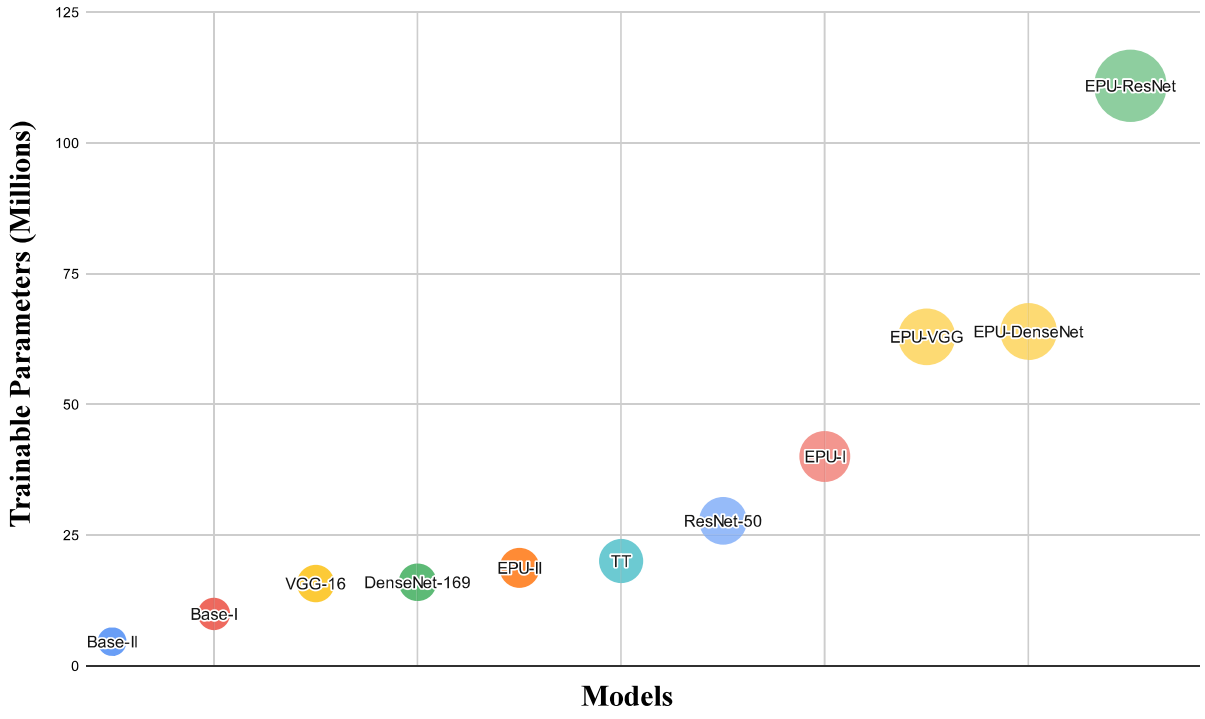


Figure 7.5. Visualization of the complexity of the compared models in terms of the number of trainable network parameters.

provide competent results even with base models of low complexity, *i.e.*, EPU_I and EPU_{II} utilize ~40 and ~19 million parameters, respectively; however, they provide higher classification performance when compared to more complex EPU-CNN models. Furthermore, the less complex EPU_{II} has comparable or even better classification performance when compared to EPU_I while it outperforms substantially $ResNet_{50}$ that is a more computationally demanding base model (~27M parameters). On the other hand, the inherently interpretable (TT) model that has a similar complexity to EPU_{II} , *i.e.*, ~20M parameters, provides the lowest overall classification performance amongst all models.

Quantitative Interpretability Analysis

To quantitatively evaluate the interpretability of the proposed framework we exploited the properties of the Banapple benchmark dataset. Banapple is suitable for this purpose because our perception of the class-related objects is directly associated with the way we categorize them, based on their visual attributes regarding color and shape (Deroy, 2013). Therefore, the subsequent task of annotating the images of Banapple did not require any domain-specific knowledge. Images of bananas and apples have distinguishable characteristics with respect to all PFMs utilized by the EPU-CNN models, *i.e.*, *light-dark*, *coarse-fine*, *blue-yellow* and *green-red*. Thus, in the case of a correct class prediction, ideally, all RSSs should trend towards the same direction, as indicated by the sign of an RSS, *e.g.*, all RSSs for an apple image should be positive, whereas for a banana image should be negative. Hence, given that the EPU-CNN models in this study use four PFMs, a ground truth, y_{int}, and predicted, $\tilde{y}_{int}$, interpretability label is expressed as follows:

$$y_{int}(y) = \begin{cases} (1,1,1,1), if\ y = 1 \\ -(1,1,1,1), if\ y = 0 \end{cases} \tag{5.10}$$

$$\tilde{y}_{int}\left(EPU_{CNN}(I;\ \eta_{\{n\}})\right) = (sign(C_1(I_1;\ \eta_1)), \dots, sign(C_4(I_4;\ \eta_4))) \tag{5.11}$$

where y is the ground truth class label of an image I, 1 and 0 denotes the apple and banana class respectively whereas $sign(C_i(I_i;\ \eta_i))$ returns the sign of an RSS. Given a set of ground truth and predicted interpretability label pairs, the interpretability accuracy a_{int}, is calculated as the average Jaccard Index(Jaccard, 1912), $J(\cdot)$, among them, as follows:

$$a_{int} = \frac{1}{k}\sum_{j=1}^{k} J\left(y_{int}(y_j), \tilde{y}_{int}\left(EPU_{CNN}(I_j;\ \eta_{\{n\}})\right)\right) \tag{5.12}$$

Table 7.2. Interpretability accuracy results of EPU-CNN models.

Metric	EPU-CNN Models				
	EPU_I	EPU_{II}	EPU_{VGG}	EPU_{ResNet}	$EPU_{DenseNet}$
a_{int} (%)	72.40 ± 1.51	72.62 ± 1.63	66.64 ± 2.21	62.90 ± 4.13	64.62 ± 2.24

EPU_I, EPU_{II} achieved the highest a_{int} with a score of 72.40±1.51% and 72.62±1.63% respectively. This means that the capacity of both EPU_I and EPU_{II} models is comparable with respect to their capacity to interpret the classification of bananas and apples. Since EPU_{II} achieves a better overall classification performance, a_{int} score and it is more computationally efficient, it has been chosen for the qualitative investigation of interpretability that is presented in the following sections.

Ablation Study

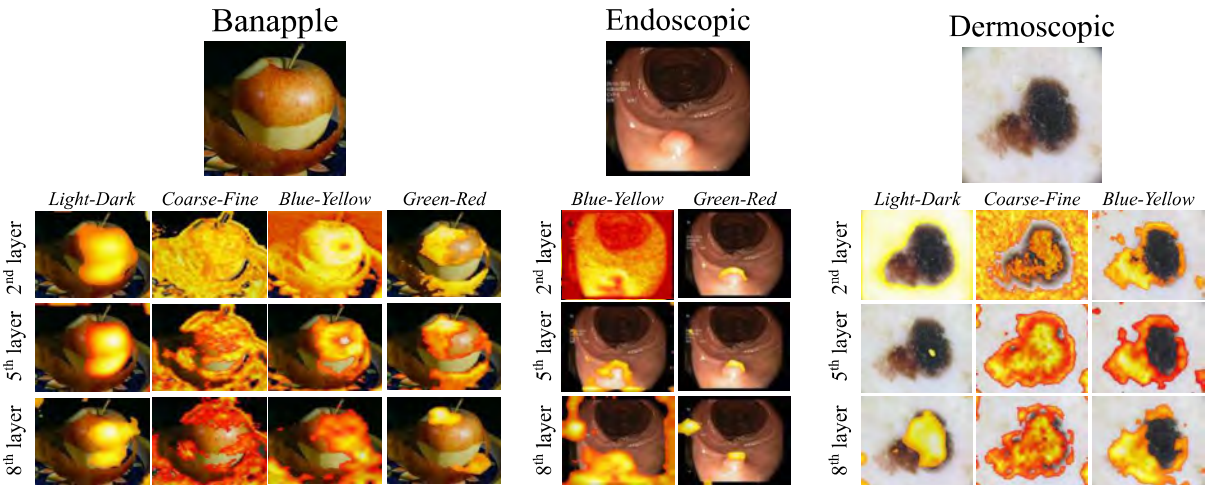


Figure 7.6. Example of PRMs generated by features maps extracted from different layers of EPU_{II}.

An ablation study was performed to determine the impact of layer selection to the construction of PRMs, using EPU_{II} as the best performing model. The feature maps estimated by 3 different layers have been chosen for the construction of the respective PRMs. Each of these layers corresponded to the last layer of each convolutional block of EPU_{II}.

Figure 7.6 illustrates indicative PRMs constructed using feature maps estimated by different convolutional layers on predictions from the Banapple, Kvasir and ISIC2019 datasets. As it can be observed, the regions identified as meaningful regarding each PFM are approximately consistent with each other regardless of the degree of abstraction that each set of feature maps encodes. However, the feature maps estimated by the intermediate 5th layer provide less noisy PRMs that highlight with more precision the areas on the input image that are estimated to be meaningful with respect to each PFM.

Qualitative Interpretability Analysis

The qualitative analysis of EPU-CNN was investigated by considering both PRMs, global and local bar-charts generated by the EPU_{II} model, for each dataset. Given a validation set of images with *a priori* known class memberships, global bar-charts are constructed by averaging the RSSs per class, as provided by each sub-network of EPU_{II}. Global bar-charts enhance the transparency of the model and reveal the overall contribution of PFMs regarding the data discrimination process.

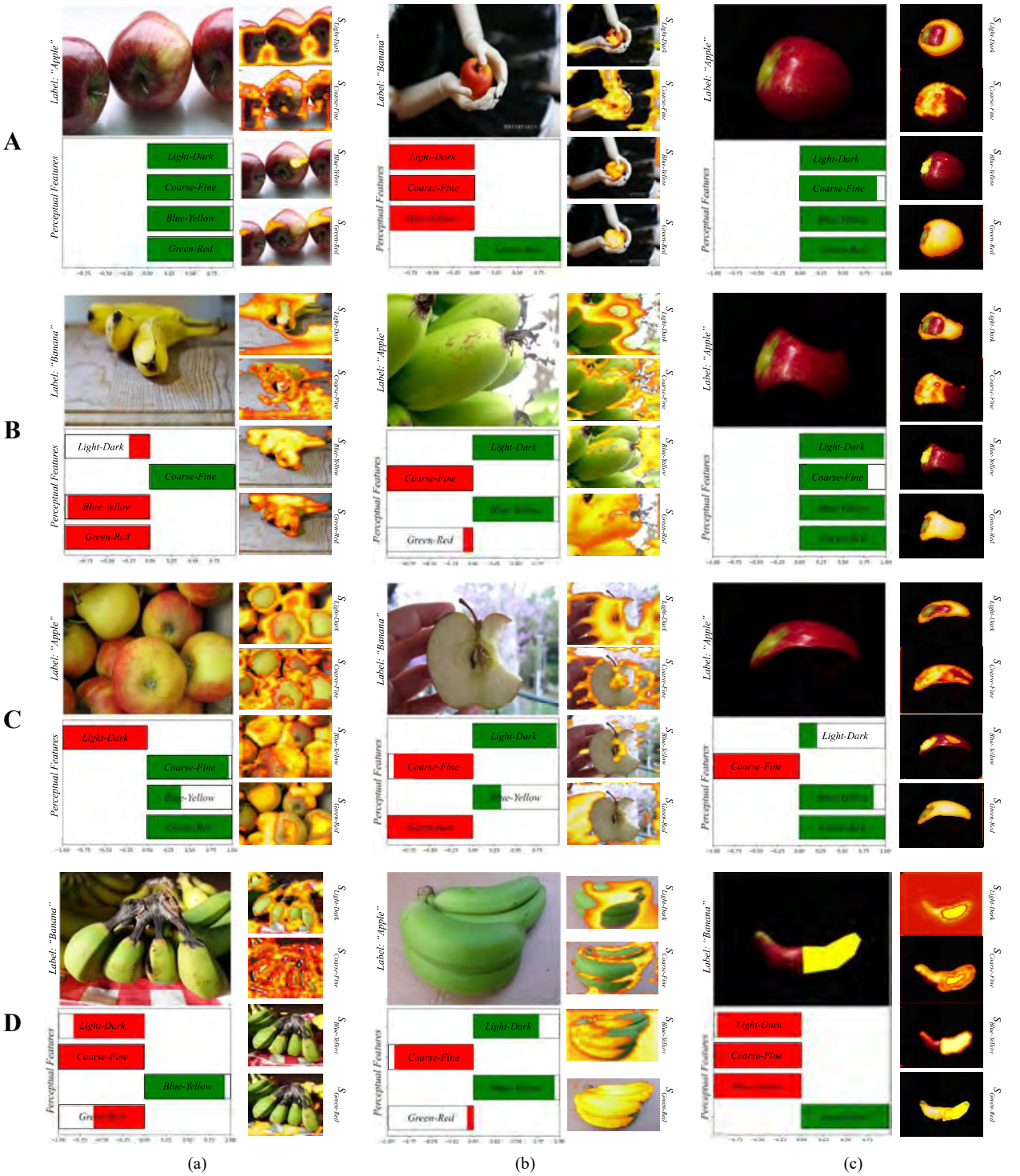


Figure 7.7. Example of local bar-charts produced by EPU_{II} on images from the Bananapple dataset. The label field indicates the predicted label. (a) Correctly classified images. (b) Wrongly classfied images. (c) Changes in the classification and its interpretation of modified images.

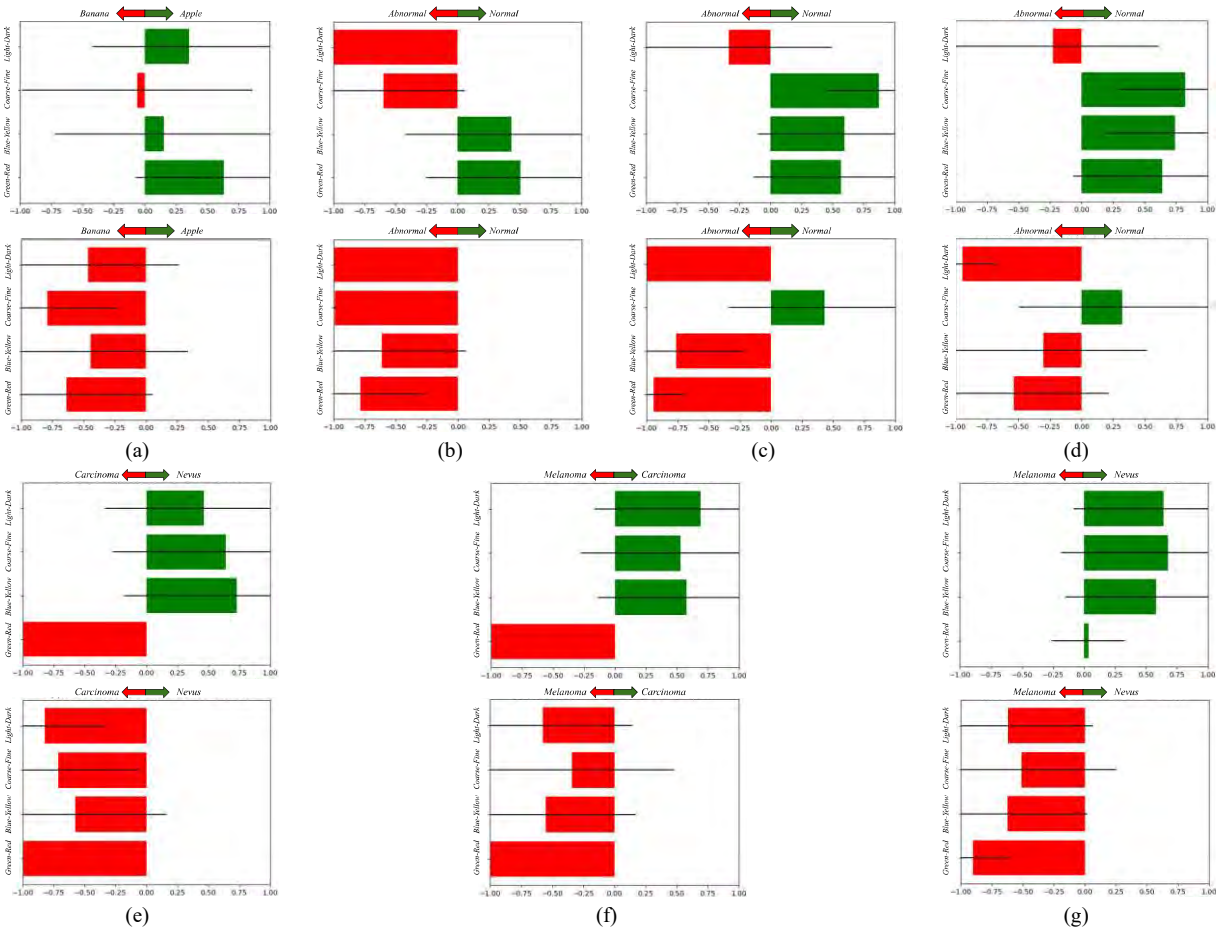


Figure 7.8. Example of dataset-wide interpretations provided by EPU-CNN on all datasets. Green (positive response) and red (negative response) bars indicating participation the 1 and 0 class respectively, and the black lines indicate the standard deviation. (a) Banapple. (b) KID. (c) MICCAI Endovis 2015. (d) Kvasir. (e-g) ISIC 2019.

In a global bar-chart, PFMs of low or high significance can be identified by their dataset-wide score, which can lead to the selection of a subset of the most informative PFMs, *i.e.,* by pruning or replacing the sub-networks corresponding to the PFMs of low significance. The respective results obtained per dataset are provided in the next paragraphs.

Banapple: The global bar-charts illustrated in Figure 7.8 (a) indicate that all the perceptual features contribute to the classification of the images. This result is in accordance with our perceptual understanding (Deroy, 2013), since apples and bananas are discriminated with respect to all PFMs considered in this study. Figure 7.7 illustrates examples of local bar-charts along with the respective PRMs of classified images. Specifically, the images presented in Figure 7.7 (a) were correctly classified by EPU_{II}, and this is reflected by the visualization of the RSSs. The PRMs of each sub-network indicate the regions of the input image which resemble the class that each RSS

suggests. For instance, in Figure 7.7 (a)-B the highlighted areas of the PRMs corresponding to *green-red* and *blue-yellow* are overlayed precisely on the class-related object, *i.e.*, the bananas. Interestingly, the difference between the *light-dark* and *coarse-fine* RSSs can be justified by the obscurity of the highlighted regions of $S_{light\text{-}dark}$ and $S_{coarse\text{-}fine}$, *i.e.*, both PRMs highlight the table. Figure 7.7 (b) illustrates wrongly classified images. Notably, each of these images have resemblances to the opposite class with respect to color and shape. For example, in Figure 7.7 (b)-A the perceptual features of *light-dark*, *coarse-fine* and blue-yellow, wrongfully guide the prediction towards the banana class (red). This can be justified since the image contains objects that share characteristics that resemble the banana class, i.e., the shape and color of the hands holding the apple. The RSS of green-red however, trends towards the apple class (green) with high magnitude, whereas the respective Sgreen-red, highlights the apple. Accordingly, the PRMs of light-dark and coarse-fine focus on the hands explaining the trend of the respective RSSs towards the banana class. Nevertheless, even though Sblue-yellow focuses on the apple, the respective RSSs indicate that the image belongs to the banana class. In Figure 7.7 (b)-B the light-dark and blue-yellow RSSs trend towards the apple class (green). The direction of these RSSs towards the incorrect class can be justified considering that the color and orientation of the bananas are not representative of their class. Accordingly, Slight-dark and Sblue-yellow focus only partially on the banana. Similarly, the shape and color from the inside of the apple in Figure 7.7 (b)-C is unusual for an apple. Hence, the coarse-fine and red-green RSSs lean towards the opposite direction.

It can be observed that the negative *red-green* and *coarse-fine* scores, have corresponding PRMs that do not focus on the class-related object, *i.e.*, they highlight regions of the hand and the background. Also, the greenish color of the bananas in Figure 7.7 (b)-D, can be descriptive for both classes (as both bananas and apples can be green), which is also expressed by the disagreement between the relative scores of the color PFMs. Interestingly, the disagreement between the *light-dark* and *coarse-fine* scores can also be justified by the highlighted regions in the respective PRMs, *i.e.,* the outline of the banana object in $S_{coarse\text{-}fine}$ and the circular region, resembling an apple in $S_{light\text{-}dark}$.

To further investigate the behavior of EPU-CNN, we have chosen an image depicting an apple (Figure 7.7 (c)-A) which was digitally processed to obtain 3 variations: *a)* to illustrate a bitten apple (Figure 7.7 (c)-B); *b*) an apple with a shape resembling that of a banana (Figure 7.7 (c)-C); and *c*) an apple resembling both the shape and color of a banana while maintaining a reddish region (Figure 7.7 (c)-D). The interpretation changes that can be observed include the following:

When the shape resembles a bitten apple the *coarse-fine* PFM is still guiding the prediction towards the apple class but with greater uncertainty (Figure 7.7 (c)-B), whereas the $S_{coarse\text{-}fine}$ discriminates the image based on its textural variations, *i.e.*, the curvature of the left side of the apple.

When the shape resembles a banana, the *coarse-fine* PFM strongly suggests that the image belongs to the banana class (Figure 7.7 (c)-C, (c)-D). The magnitude of *light-dark* RSS has also changed, but still trends towards the apple class. The $S_{light\text{-}dark}$ and $S_{coarse\text{-}fine}$ appear to contribute to the

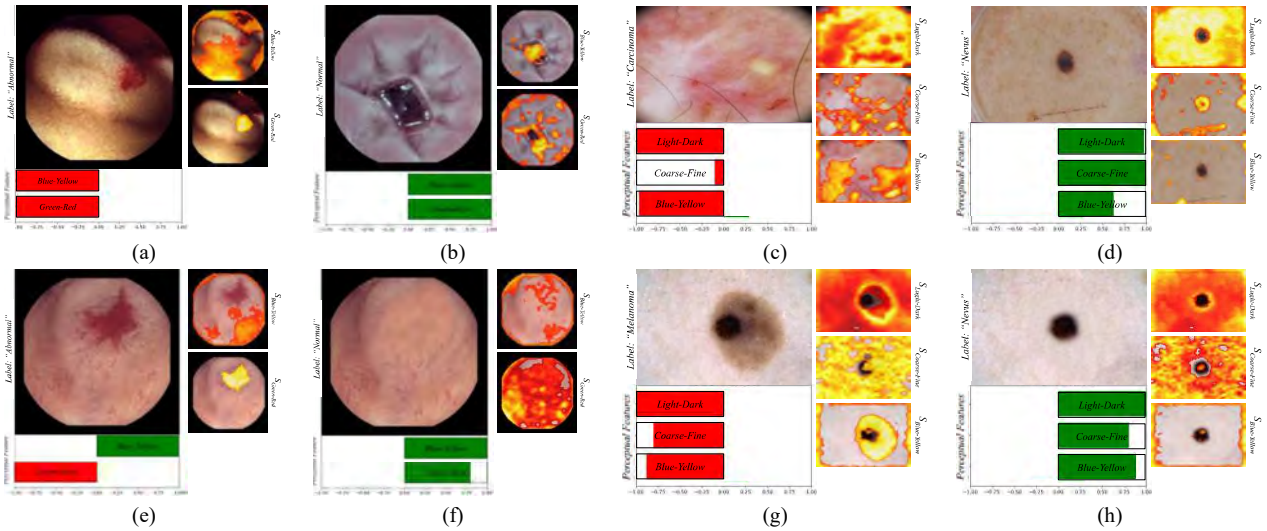


Figure 7.9. Example of EPU-CNN interpretations, as generated by EPU_{II}, on biomedical images. The label field indicates the predicted label. (a) Abnormal and (b) normal endoscopic image; (c) Carcinoma and (d) (normal) nevus skin lesion; (e) Abnormal endoscopic image and (f) modification of (e) to resemble a normal endoscopic image; (g) Melanoma skin lesion and (h) modification of (g) to resemble nevus.

segregation the depicted object; however, only the RSS of *coarse-fine* PFM suggests the opposite class, indicating that is more sensitive to shape variations.

When the yellow region is added, the *light-dark* and *green-yellow* PFMs guide the prediction to the banana class. However, the *green-red* RSS trends towards the apple class. As expected, $S_{blue\text{-}yellow}$ and $S_{green\text{-}red}$ focus on the yellow and red segments of the object respectively (Figure 7.7 (c)-D). This justifies the trend of each PFM towards either class, *i.e.*, yellow and red are representative colors of banana and apple class respectively.

These interpretations reveal that the *coarse-fine* PFM enables the respective sub-network to respond to different shape variations and infer relevant decisions. In addition, the color related PFMs, *i.e.*, *blue-yellow* and *green-red*, are very sensitive to the class-related colors and it is clearly reflected both in the respective PRMs and RSSs. When both the class-related colors, *i.e.*, *yellow* and *red*, cooccur in the image, the *blue-yellow* and *green-red* PFM guide the prediction towards the banana and apple class respectively.

Endoscopic Datasets: The experiments showed that EPU_{II} tend to discriminate normal from abnormal images of the endoscopic datasets mainly based on the *blue-yellow* and *green-red* PFMs. This is illustrated in the respective global bar-charts (Figure 7.8 (b-d)). As it can be observed, the *light-dark* and *coarse-fine* are biased towards a specific class, in all endoscopic datasets. On the other hand, the chromatic PFMs are the main contributors to the correct classification predictions. This finding is in accordance with the literature since it has been proven that color has a leading role in finding abnormalities in the gastrointestinal tract (Dimitris K Iakovidis & Koulaouzidis, 2014).

An example of local bar-charts visualizing the prediction interpretations of EPU-CNN on endoscopic images is presented in Figure 7.9 (a, b). Since the light-dark and coarse-fine features are not informative, only the color related PFMs were considered. Figure 7.9 (a, b) illustrate correctly classified endoscopic images of both the normal and abnormal class. In the case of the image depicting an abnormality (Figure 7.9 (a)), the Sgreen-red indicates that the focus of the sub-network that corresponds to the *green-red* PFM focuses on the abnormality, *i.e.*, blood, whereas $S_{blue\text{-}yellow}$ focuses on normal tissue and only partially on the abnormal region.

To assess the behavior of EPU_{II} in a more controlled way in the endoscopic datasets, we proceeded to digitally process an endoscopic image and create different conditions for their classification to the normal and the abnormal classes. Indicative examples are presented in Figure 7.9 where the abnormal region of Figure 7.9 (e) is removed, resulting in the synthetic image of Figure 7.9 (f). The qualitative result of this process, considering only the chromatic PFMs, is reflected in the RSSs and the PRMs of Figure 7.9 (f). Specifically, it can be noticed that by replacing the abnormal region with normal tissue, the RSS of the *green-red* PFM shifts from trending towards the abnormal class (red) to the normal class (green). Furthermore, the RSS of the *green-red* PFM, in the absence of an abnormality, indicate that the respective subnetwork focuses on normal tissue.

Dermoscopic Datasets: The evaluation of the interpretability of EPU_{II} on the dermoscopic datasets revealed that all the PFMs participate actively in the classification process with an exception to *green-red* PFM that appears biased towards either the Melanoma or Carcinoma class on all trials (Figure 7.8 (e-g)). This is an indication that the PFM of *green-red* is not informative to the network regarding the classification of dermoscopic images. Furthermore, as it can be observed in Figure 7.8 (e-g) the classification process of EPU_{II} is relying on both chromatic and textural cues (*i.e.*, *blue-yellow, light-dark* and *coarse-fine*) that are also considered by the ABCD rule of skin lesion classification to assess the malignancy of a lesion (Nachbar et al., 1994).

An example of local bar-charts of classified dermoscopic images are illustrated in Figure 7.9 (c, d). The local bar-chart includes the most informative PFMs, *i.e.*, *light-dark*, *coarse-fine* and *blue-yellow*. In Figure 7.9 (c, d) all RSSs are trending, correctly, towards the abnormal (carcinoma, red) and normal (nevus, green) class respectively. In the case of the carcinoma (Figure 7.9 (c)), $S_{light\text{-}dark}$ focuses on the entirety of the image, whereas $S_{coarse\text{-}fine}$ and $S_{blue\text{-}yellow}$ focuses on regions with color variations, *e.g.*, on the yellow spot and little cuts on the lest and bottom side of the image respectively. In the case of the nevus (Figure 7.9 (d)), $S_{light\text{-}dark}$ and $S_{coarse\text{-}fine}$ isolate the lesion by segregating it from the rest of the image, either by focusing on it or around it, whereas $S_{blue\text{-}yellow}$ indicates only a slight attention of the network to the lesion. Similarly, to the other datasets, we proceeded to digitally modify the image of Figure 7.9 (g) that illustrates a melanoma to resemble a nevus. The modification was implemented according to the rule-based diagnostic criteria expressed by the ABCD rule(Nachbar et al., 1994); in detail, we removed the part of the lesion that introduced color variation on the same mole and obtained a more symmetrical shape. The qualitative results of

this process are illustrated in Figure 7.9 (h), where it is shown that after the modification all the RSSs trend towards the nevus class (green). Furthermore, the $S_{blue\text{-}yellow}$ PRM, in the absence of an abnormal region, indicate that the respective subnetwork does not focus on the skin lesion. The $S_{light\text{-}dark}$ and $S_{coarse\text{-}fine}$ PRMs seem to maintain a similar behavior with the unmodified image.

Comparison with State-of-the-Art Interpretable Methods

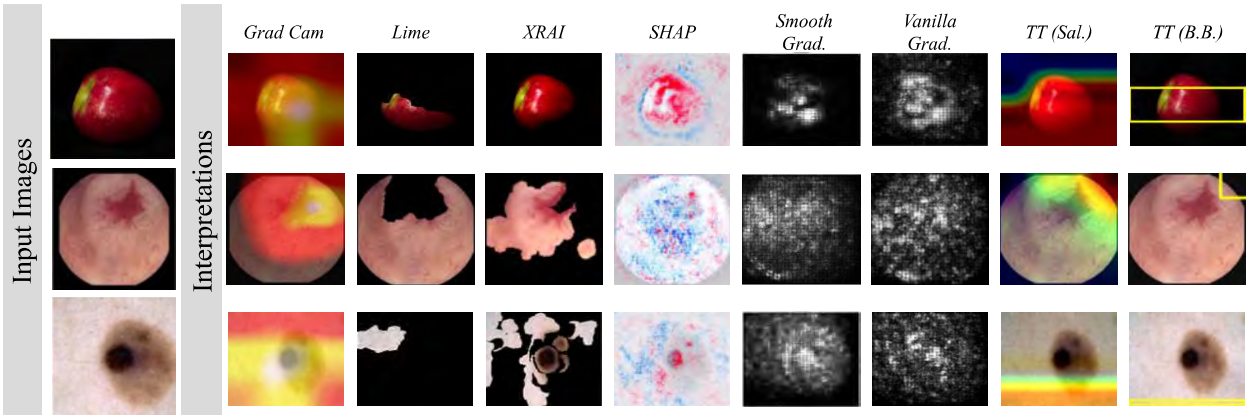


Figure 7.10. Example of CNN interpretations provided by various methodologies.

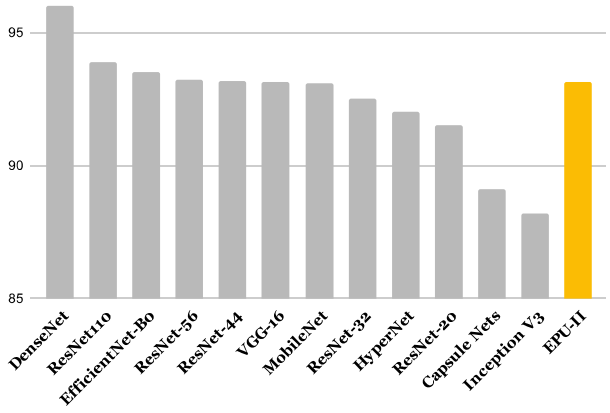


Figure 7.11. Classification performance in terms of accuracy on the CIFAR-10 dataset.

Even though there is an increasing research interest regarding the interpretation of CNNs, there is still not a standard procedure to evaluate and compare the interpretable output. Nevertheless, a qualitative comparison can reveal strengths and weaknesses of such methods. In this study, the interpretations that EPU-CNN provides are qualitatively compared to seven methodologies that have been proposed to interpret CNNs and have been also widely used in the literature. These methods provide saliency maps indicating regions or points on the input image that are estimated to be crucial for a prediction inferred by a CNN. In detail, six *post-hoc* methodologies, namely, Grad-CAM (Selvaraju et al., 2017), LIME (Ribeiro et al., 2016), XRAI (Kapishnikov, Bolukbasi, Viégas, & Terry, 2019), Shapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017b), Smoothgrad (Smilkov et al., 2017) and Vanilla Gradients (Simonyan, Vedaldi, & Zisserman, 2013), as well as one inherently interpretable model (C. Chen et al., 2019) (TT) were utilized in this evaluation. The *post-hoc* methodologies were applied on the CNN models that achieved the highest performance on each dataset according to Table 7.1, whereas TT was trained on each dataset from scratch. All the methods provide interpretations in the form of saliency maps while TT can also provide bounding

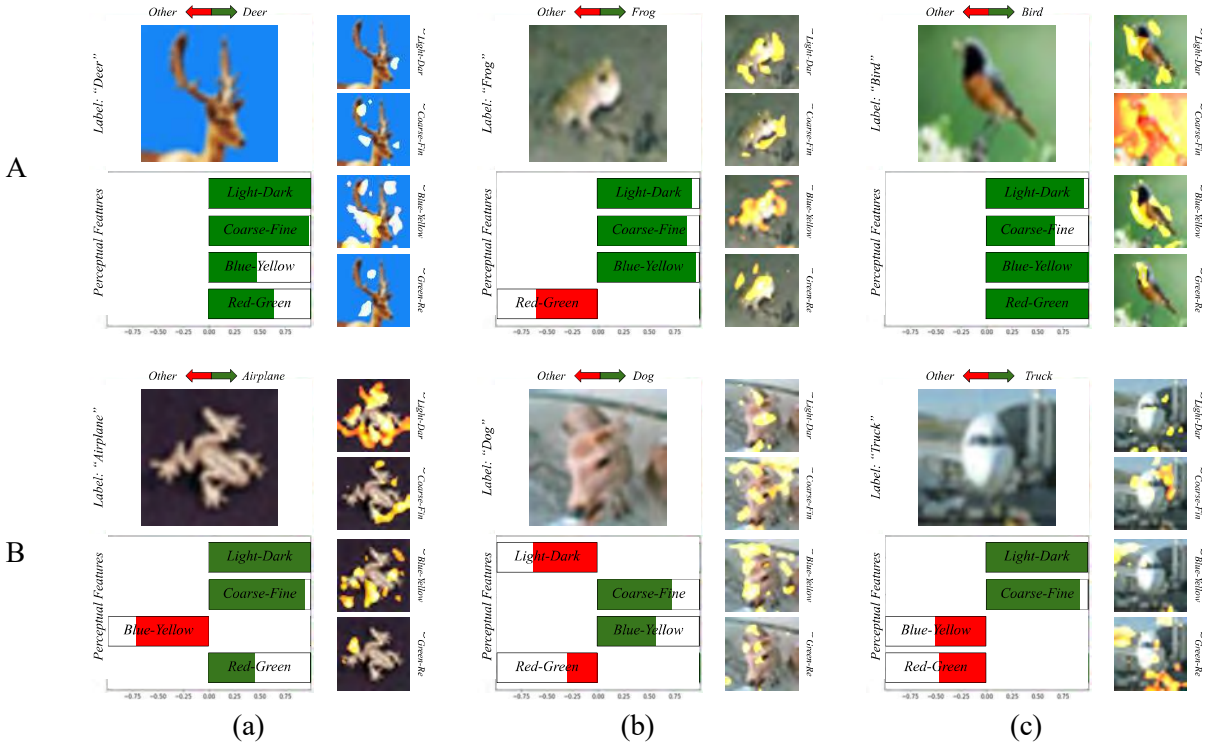


Figure 7.12. Example of EPU-CNN interpretations, as generated by EPU_{II}, on images of the CIFAR-10 dataset. The label field indicates the predicted label. Row A and B illustrate interpretations of correct and wrong prediction, respectively.

boxes that specify discreetly the estimated region of interest. These methods were selected since they can render CNN models interpretable without the need for training on datasets specifically annotated for interpretable learning, *e.g.*, with annotation regarding the concepts that are depicted on images. Figure 7.10 summarizes the interpretations provided by each method on exemplary images that are presented in Figure 7.7 and Figure 7.9. All the images have been correctly classified by the respective models that were used. In detail, only XRAI and SHAP were successful at highlighting regions of interest on the images that can be regarded crucial for classification, *i.e.*, areas of the apple, the skin lesion and blood depicted in the endoscopic image. The gradient-based interpretation approaches, *i.e.*, Grad-CAM, Smoothgrad and Vanilla Grad., also revealed that the respective CNN models focus on image regions that can be regarded meaningful; nevertheless, the fuzziness of their visualization makes the communication of their interpretations difficult to comprehend. On the other hand, EPU-CNN can provide different visualizations, that highlight the most relevant regions with respect to each PFM as it was estimated by the layer activations of each subnetwork. This can also be expressed quantitatively since the RSSs indicate the degree to which each highlighted region affects the classification result. Furthermore, the dataset-wide plots that can be constructed by using an

EPU-CNN model give insights regarding which PFMs are important for classifying the images of a particular dataset. To the best of our knowledge no other interpretation approach can incorporate all this information to its explanations and simultaneously be applied on non-specialized datasets, *e.g.*, datasets where each image is annotated only with respect to their class membership.

Figure 7.12 illustrates interpretations provided by EPU_{II} on predictions of images on the CIFAR-10 dataset. For consistency with the interpretations of the binary settings, the respective illustrations depict how each PFM drives a prediction towards either the predicted (green) or any other class (red). The rows A and B of Figure 7.12 illustrate interpretations of correct and wrong classifications on images included in the CIFAR-10 dataset, respectively. As it can be observed the PRMs generated by the EPU_{II} on the interpretations presented in Figure 7.12-A highlight the object of interest with more precision when compared to the wrongly classified images (Figure 7.12-B). For example, in Figure 7.12 (b)–A all the PRMs highlight regions of the frog, and the image has been correctly classified. On the other hand, in Figure 7.12 (a)-B the PRMs mainly highlight regions around the frog and the respective image is misclassified to the airplane class, based on all the PFMs except from the *blue-yellow*. Furthermore, it can be noticed that the respective RSSs behave similarly, *i.e.*, in Figure 7.12 (c)-A the PRM of *coarse-fine* highlights the whole image and it does not focus solely on the bird. Accordingly, the respective RSS of *coarse-fine* has a smaller magnitude towards the correct class than the rest of RSSs, which, based on the respective PRMs consider mainly the region of the bird. Moreover, in Figure 7.12 (b)-B the image that belongs to the deer class is misclassified as a dog. As it can be noticed, the PRMs of *coarse-fine* and *blue-yellow* are mainly focusing on the head region of the deer and the respective RSSs drives the prediction towards the dog class. This can be attributed to the fact that the particular deer does not seem to have horns, and it has a color pattern that matches that of the dog class. This can be further substantiated by observing the image of Figure 7.12 (a)-A. This image depicts a deer that has been correctly classified by the EPU-CNN model. As it can be noticed, the PRMs of *coarse-fine* and *blue-yellow* highlight the head region of the deer where the horns are present. Finally, the image of Figure 7.12 (c)-B that depicts an airplane is classified to the truck class based on the PFMs of *light-dark* and *coarse-fine*. As it can be noticed, all the PRMs focus on the body of the airplane, and on the wheels, whereas no PRM focuses on the wings. Therefore, the respective RSSs of *light-dark* and *coarse-fine* drive the prediction with a higher magnitude towards the truck class. Figure 7.11 presents a comparison in terms of classification accuracy among EPU_{II} (orange bar) and other state-of-the-art CNN models (G. Huang et al., 2017),(He et al., 2016),(Simonyan & Zisserman, 2014),(Ha, Dai, & Le, 2016), (Sabour, Frosst, & Hinton, 2017) (gray bars) on the CIFAR-10 dataset. EPU_{II} achieved an accuracy score of 93.31% which is comparable or better than the other models considered. However, a major advantage over the other models is that the EPU-CNN model can provide interpretations regarding the classification outcome.

7.2 Discussion and Future Work

In section 7.1 a novel, generalized framework, called EPU-CNN is proposed. EPU-CNN provides a guideline for the development of interpretable CNN models, inspired by GAMs. A model, designed according to EPU-CNN framework, consists of an ensemble of sub-networks with a base CNN architecture that is trained as one. The proposed framework can be used to render conventional CNN model interpretable, by using it as a base model. Each sub-network receives as input a different PFM of an input image, chosen according to the literature of cognitive science and human perception. EPU-CNN is designed in such way enabling human-friendly interpretations of its classification results based on the utilized perceptual features. The interpretations provided EPU-CNN are in the form of RSSs that quantify the resemblance of a perceptual feature to a respective class. These interpretations are complemented by PRMs indicating the image regions where the network focuses to infer its interpretable decisions. Furthermore, EPU-CNN provides spatial expression of an explanation on the input image. Thus said, the most important conclusions of this study can be summarized as follows:

EPU-CNN models satisfy the need for interpretable models based on human perception, *i.e.*, the proposed framework is able to provide interpretations in accordance with human perception and cognitive science, *e.g.*, EPU-CNN classifies endoscopic images based on the chromatic perceptual features. Unlike other inherently interpretable CNN methodologies (Q. Zhang et al., 2018, 2019), the classification performance of EPU-CNN models is not affected by their capacity to provide interpretations. In fact, the results obtained from the comparison of EPU-CNN models with respective non-interpretable CNN models, show that their performance is better or at least comparable to that of the non-interpretable models. In addition, when an image is modified with respect to a perceptual feature, *e.g.*, color, the interpretations derived from the EPU-CNN model change accordingly both on natural and biomedical images (Figure 7.7 and Figure 7.9). Moreover, since EPU-CNN is a generalized framework, it provides a template for the development of interpretable CNNs that fulfill the requirements imposed by current legislations regarding the commercial applicability of ML models.

Most inherently interpretable models that have been proposed in the literature, can only be applied on datasets that are further annotated with respect to human-understandable concepts illustrated in each image, which results in limitation regarding their applicability (Barbiero et al., 2022; Bau et al., 2017). The PFM selection of an EPU-CNN model can be considered as a less demanding and time-consuming procedure when compared to the annotation of huge datasets with the human-understandable concepts. Additionally, since the selection of the textural and color perceptual features, that are based on the 2D DWT and *Lab*, respectively, is empirical, as a future work we intend to automate the PFM selection towards a direction that minimizes human intervention and is more compatible with the principles of deep learning.

An aspect of EPU-CNN that can be considered as a limitation of the proposed framework, is the manual selection of PFMs. The process of the selection, however, enables the user to leverage

specific PFMs that are relevant to a particular application and as a result to acquire meaningful interpretations and insights regarding the internal process of an EPU-CNN model. For example, in the case of endoscopic images, the EPU-CNN models considered only the PFMs of color as more important which is in accordance with the respective literature. As future work, we intend to investigate ways for the automated selection of PFMs or the generation of PFMs with certain attributes by each subnetwork.

Chapter 8
Conclusions and Future Research Directions

The domain of Machine Perception and Computer Vision is full of challenges. The research conducted in the context of this dissertation contributed to the advancement of science by investigating novel methodologies to cope with visual saliency prediction, obstacle detection, visual size measurements, 3D model reconstruction and refinement, as well as perceptually interpretable ML. These methodologies were investigated in the context of various applications with societal impact, including assistive systems for the navigation of visually impaired individuals based on visual cues, and medical decision support systems for detection.

The spatial regions attracting the attention of human observers is a significant indicator of how humans comprehend images, i.e., where the important regions of the image are placed, or where an action or event occurs (Bylinskii et al., 2016). Given enough data, an ML model can be trained to predict these locations. Henceforth, Sections 3.1 and 3.2 present methodologies for detecting gaze patterns in a number of normal and pathological WCE image categories. According to the results, the enhanced architecture that employs an additional convolutional reconstruction block combined with a post-refinement step outperformed both the basic model and the state-of-the art methods that have been proposed for gaze prediction in biomedical images. Furthermore, in section 3.2 the utilization of a novel co-operative training procedure is proposed. This training scheme incorporates two models trained jointly in a co-operative manner. Each network is dedicated to a specific task; one network focuses on predicting salient maps whereas the other utilizes the saliency maps along with their respective input images for classification. Hence, their co-operation considers both the saliency map accuracy and its effect on the classification of abnormalities. A model trained according to this scheme outperforms other relevant gaze prediction methodologies in terms of accuracy of visual saliency prediction. To the best of our knowledge, these are the first deep learning models that have been developed for the gaze estimation of physicians on biomedical images.

The monocular salient object detection approach presented in section 3.3 focuses on natural images and it is designed to cope with the dependency of previous approaches on sensor-based depth information. The novel depth-aware, two-branch CNN salient object detection methodology, referred to as MonoSOD, utilizes predicted depth maps instead of depth information deriving from specialized sensors. The performance of MonoSOD indicates that the incorporation of predicted depth, can provide comparable performance for SOD with that of sensor-based approaches. Additionally, MonoSOD outperforms the state-of-the-art D^3Net when both utilize predicted depth while maintaining a smaller architectural complexity.

A novel depth-aware obstacle detection method, based on fuzzy sets and visual saliency prediction, was presented in section 4.1. This method has been extended with the incorporation of personalized parametrization along with a simple ground plane removal approach and it was subsequently applied for computer-assisted navigation, as described in 4.2. The personalization aspects of the methodology presented in section 4.2, alongside with the ground plane removal, provides a significant lower false alarm rate when compared to its preliminary counterpart (section 4.1). At the same time, both methodologies outperformed in terms of obstacle detection accuracy approaches that are solely based on depth information. Furthermore, the use of just an RGB-D sensor results to the minimization of resources that would otherwise require the integration, fusion, and synchronization of multiple sensors. Additionally, the proposed RGB-D obstacle detection methodology (section 4.2) has been integrated in an assistive system for the navigation of VCPs, named ENORASI. To evaluate the efficacy of the proposed method, in terms of obstacle detection and avoidance, a user evaluation study has been conducted. In this study, VCPs were wearing the ENORASI system, employing the proposed obstacle detection methodology, to navigate outdoors (Mitsou et al., 2022). The results indicate that the proposed approach can efficiently assist VCPs to safely avoid obstacles in outdoor environments.

An alternative obstacle detection methodology, based on self-supervised learning, was presented in section 4.4. The results obtained by the self-supervised model in the context of obstacle detection provide evidence this type of training can lead to a less complex, faster, and scalable obstacle detection method. This approach simplifies the obstacle detection methodology of 4.2 and is capable of efficiently detecting regions containing possible high-risk obstacles given only an RGB image as input. It should be noted that the obstacle detection methods proposed in section 4 are generic and they can also be used in other application, such as navigation of autonomous vehicles and robotic agents in general.

Regarding the visual measurement methodologies that have been described in section 5, the respective evaluation study suggests that they can successfully be employed for the measurement of objects both in everyday applications, *in-the-wild*, and medical applications. In the case of natural images, since the proposed approach requires only a single image and knowledge about the intrinsic parameters of the camera along with its approximate position, it can be used to simplify systems that currently use stereo camera rigs or specialized sensors, *e.g.*, LIDAR. In the context of size measurements during endoscopy and particularly WCE examination, *i.e.*, *in-vivo*, such

methodologies are crucial. The simplification of the device requirements, especially in the case of WCE, leads to more efficient designs of medical devices. Size measurements in WCE images has been an under-research subject, and prior studies usually focused on using multiple images and visual odometry that some-times, considering the kinematics of the capsule inside the bowel, is a very tedious task (Dimitris K Iakovidis, Dimas, et al., 2018). Hence, the development of single-image approaches can be employed to overcome these difficulties providing a more robust way for the assessment of the size of lesions or other findings to aid the decision-making process of physicians.

With respect to the reconstruction of 3D models given a sparse 3D object representation, the proposed unsupervised approach described in 6.1 provides a simple and effective method for tackling such problems. This is achieved by leveraging the universal approximation capabilities of neural networks along with a novel activation function, named WaveShaping (WS), that is used to train a model to approximate, perceive and represent a function describing a 3D model. Once such a model is trained, it can be used to interpolate data points that are missing from the original sparse 3D representation of the model and produce a finer version that can be used for the development of digital twins. This is important since 3D representations of organs can be refined and used for more accurate and precise *in-silico* trials.

To cope with the problem of interpretability in CNNs, this dissertation introduces EPU-CNN framework (section 7.1). This novel framework provides a guide for the construction of perceptually interpretable CNN models. The EPU-CNN models can be regarded as a new class of models tasked to tackle both binary and multi-class interpretable classification tasks in the domain of CV. The classification results obtained from EPU-CNN models show that they perform better or at least on par when compared to their non-interpretable counterparts. In addition, when an image is modified with respect to a perceptual feature, *e.g.*, color, the interpretations derived from the EPU-CNN model change accordingly both on natural and biomedical images. Another important feature of EPU-CNN models is that they can provide interpretations in terms of salient regions that correspond to perceptual attributes of an image along with their quantified contribution to the final prediction. On the other hand, other interpretable methods are capable of only providing image regions that the model finds important without quantifying their contribution to the prediction or associating them with a perceptual image component. Furthermore, EPU-CNN models fulfill the requirements imposed by current legislations regarding the commercial applicability of ML models. Unlike other methods, EPU-CNN does not require training on predefined concepts to support the interpretations regarding its classification results and can be applied on any dataset. Its interpretations are based on perceptual image components, which, according to cognitive science, are also utilized by humans to perform classification-related tasks (Greisdorf & O'Connor, 2002).

The methodologies introduced in this dissertation cope with several known problems in the field of Machine Perception and Computer Vision; yet, they still have some limitations summarized below:

i. The predicted depth used in the context of MonoSOD enables the model to provide comparable performance to state-of-the-art SOD methods but did not manage to outperform the utilization of sensor-based depth.

ii. The self-supervised approach proposed in section 4.4 is able to approximate the performance to its supervisor but needs further improvement to be able to replace it in real world applications

iii. Even though the single-image methodologies presented in sections 5.1 and 5.2 outperform other measurement approaches, they require the fine tune of a set of parameters that can be considered as a limitation. Furthermore, currently single image measurement methodologies measure the dimensions of an object in terms of linear segments within a bounding box surrounding the target object and they are not modeled to estimate the size of complex 3D surfaces.

iv. Regarding EPU-CNN (section 7.1), the manual selection of PFM relies to the user for the correct assessment of the required information for the solution of a problem. The lack of automated PFM selection can lead to the incorporation of irrelevant PFMs that may lead to the decreased performance of the model or additional trials.

Considering the limitations listed above and the constantly increasing pace that the DL field is advancing, this doctoral dissertation concludes by suggesting future research directions:

i. Regarding saliency prediction, the optimization of depth maps to outperform sensor-based models is a challenging research direction. The results obtained by MonoSOD should motivate studies to explore whether the pixel-level precision of depth values or a rougher representation of the depth in a scene is more beneficial for a model. Furthermore, different representations of complementary information should be examined to determine their effect in the performance of saliency prediction methodologies.

ii. More approaches should focus on the improvement of performance concerning self-supervised obstacle detection approaches. The results provided by our approach have shown that this direction is promising however more improvements either by using more advanced network architectures or more data incorporated in the training must be considered. Furthermore, the incorporation of predicted depth as input to the model trained for obstacle detection is a promising approach for future work.

iii. Training DL models using self-supervised approaches should be further investigated and incorporated in various methodologies where automated supervision provided by other algorithms is possible. As it was presented in section 4.4, training a model to learn another

algorithm using a self-supervised training scheme can lead to a simpler and time-efficient solution.

iv. Single-image measurement approaches should be further researched and optimized. The proposed approaches can be further improved with the introduction of an automated approximation procedure of the parameters regarding the virtual grid generation. It would be interesting as a future work to develop methods that automatically estimate these parameters based on the intrinsic parameters of the camera and the application environment, e.g., measurements in the gastrointestinal tract and in outdoors environments require a sparser and denser grid, respectively. Another future direction regarding the improvement of single-image measurement methodologies is the improvement of their model for estimating the size of complex 3D shapes instead of just the height, width and length of an object. This can be achieved with the utilization of advanced segmentation algorithms that can precisely define the boundaries of an object.

v. Motivated by the study presented in section 5 future research directions include further exploration of periodic activation functions and how different types of parameterizations affect the performance of the reconstruction model. Moreover, the robustness in terms of performance of WS function given datasets with low number of samples, its employment of other classes of problems, such as classification, image segmentation *etc.*, fields should be considered as a direction for future research.

vi. In the context of EPU-CNN (section 7.1) a future research direction includes the development of methodologies that can automatically determine which Perceptual Feature Maps (PFMs) are considered important for developing an interpretable EPU-CNN model that can tackle a problem efficiently. These automated approaches can be either embedded to the model, i.e., an additional convolutional block that generates a PFM representation according guided by prior knowledge regarding various image components, e.g., color, or independent modules. Another interesting direction if the employment of EPU-CNN framework in the context of a novel direction towards interpretable image mapping problems such as, interpretable depth prediction, image segmentation, saliency prediction etc.

APPENDIX A

List of Publications Resulted from this Dissertation

Journals

Dimas, G., & Iakovidis, D. K. (2022). Virtual Grid Mapping for Visual Size Measurements. IEEE Transactions on Instrumentation and Measurement. (revised, waiting response)

Dimas, G., Cholopoulou, E., & Iakovidis, D. K. (2022). E Pluribus Unum Interpretable Convolutional Neural Networks. Pattern Recognition. (under review)

Iakovidis, D. K., Ooi, M., Kuang, Y. C., Demidenko, S., Shestakov, A., Sinitsin, V., …, Dimas, G., ... & Gao, R. X. (2021). Roadmap on signal processing for next generation measurement systems. Measurement Science and Technology, 33(1), 012002.

Ntakolia, C., Dimas, G., & Iakovidis, D. K. (2020). User-centered system design for assisted navigation of visually impaired individuals in outdoor cultural environments. Universal Access in the Information Society, 1-26.

Dimas, G., Bianchi, F., Iakovidis, D. K., Karargyris, A., Ciuti, G., & Koulaouzidis, A. (2020). Endoscopic single-image size measurements. Measurement Science and Technology, 31(7), 074010.

Dimas, G., Diamantis, D. E., Kalozoumis, P., & Iakovidis, D. K. (2020). Uncertainty-aware visual perception system for outdoor navigation of the visually challenged. Sensors, 20(8), 2385.

Conferences and Book Chapters

Dimas, G. & Iakovidis, D. K. (2023). Co-Operative CNN for Visual Saliency Prediction on WCE Images. In ICASSP 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6623-6627). IEEE. (submitted)

Triantafyllou, G., Dimas, G., Kalozoumis, P., & Iakovidis, D. K. (2022, August). Reconstruction of Cultural Heritage 3D Models from Sparse Point Clouds Using Implicit Neural Representations. In International Conference on Pattern Recognition. Springer, Cham. (to appear)

Iakovidis, D. K., Diamantis, D., Dimas, G., Ntakolia, C., & Spyrou, E. (2020). Digital enhancement of cultural experience and accessibility for the visually impaired. In Technological Trends in Improved Mobility of the Visually Impaired (pp. 237-271). Springer, Cham.

Vasilakakis, M., Sovatzidi, G., Dimas, G. & Iakovidis, D. K. (2022). Towards the Interpretation of Convolutional Neural Networks for Image Classification Using Fuzzy Sets. IEEE Systems, Man, and Cybernetics Magazine. (to appear)

Kalozoumis P. G., Dimas G., Triantafyllou G. & Iakovidis D. K. (2022). A Framework for the Development of Intestinal Digital Twins Integrating Machine Learning and Multiphysics Modelling. 2022 Virtual Physiological Human Conference, Sept. 6th-9th, Porto, Portugal

Mitsou, A., Koutsiou, D. C. C., Diamantis, D. E., Psallidas, T., Dimas, G., Vasilakakis, M., ... & Iakovidis, D. K. (2022, June). ENORASI Assistive Computer Vision-based System for the Visually Impaired: A User Evaluation Study. In Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments (pp. 668-677).

Dimas, G., Cholopoulou, E., & Iakovidis, D. K. (2021, August). Self-Supervised Soft Obstacle Detection for Safe Navigation of Visually Impaired People. In 2021 IEEE International Conference on Imaging Systems and Techniques (IST) (pp. 1-6). IEEE.

Gatoula, P., Dimas, G., Iakovidis, D. K., & Koulaouzidis, A. (2021, June). Enhanced CNN-Based Gaze Estimation on Wireless Capsule Endoscopy Images. In 2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS) (pp. 189-195). IEEE.

Dimas, G., Gatoula, P., & Iakovidis, D. K. (2021, May). MonoSOD: Monocular Salient Object Detection based on Predicted Depth. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4377-4383). IEEE.

Dimas, G., Iakovidis, D., & Koulaouzidis, A. (2019, October). MedGaze: Gaze Estimation on WCE Images Based on a CNN Autoencoder. In 2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE) (pp. 363-367). IEEE.

Dimas, G., Ntakolia, C., & Iakovidis, D. K. (2019, May). Obstacle detection based on generative adversarial networks and fuzzy sets for computer-assisted navigation. In International Conference on Engineering Applications of Neural Networks (pp. 533-544). Springer, Cham.

Other Related Publications of the Author

Journals

Iakovidis, D. K., Dimas, G., Karargyris, A., Bianchi, F., Ciuti, G., & Koulaouzidis, A. (2018). Deep endoscopic visual measurements. IEEE Journal of Biomedical and Health Informatics, 23(6), 2211-2219.

Koulaouzidis, A., Iakovidis, D. K., Yung, D. E., Mazomenos, E., Bianchi, F., Karagyris, A.,…, Dimas, G., ... & Ciuti, G. (2018). Novel experimental and software methods for image reconstruction and localization in capsule endoscopy. Endoscopy international open, 6(02), E205-E210.

Dimas, G., Spyrou, E., Iakovidis, D. K., & Koulaouzidis, A. (2017). Intelligent visual localization of wireless capsule endoscopes enhanced by color information. Computers in biology and medicine, 89, 429-440.

Dimas, G., Iakovidis, D. K., Karargyris, A., Ciuti, G., & Koulaouzidis, A. (2017). An artificial neural network architecture for non-parametric visual odometry in wireless capsule endoscopy. Measurement Science and Technology, 28(9), 094005.

Conferences

Dimas, G., Iakovidis, D. K., Ciuti, G., Karargyris, A., & Koulaouzidis, A. (2017, June). Visual localization of wireless capsule endoscopes aided by artificial neural networks. In 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS) (pp. 734-738). IEEE.

Iakovidis, D. K., Dimas, G., Karargyris, A., Ciuti, G., Bianchi, F., Koulaouzidis, A., & Toth, E. (2016, October). Robotic validation of visual odometry for wireless capsule endoscopy. In 2016 IEEE International Conference on Imaging Systems and Techniques (IST) (pp. 83-87). IEEE.

Bibliography

Achlioptas, P., Diamanti, O., Mitliagkas, I., & Guibas, L. (2018). Learning representations and generative models for 3d point clouds. *International conference on machine learning* (pp. 40–49). PMLR.

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*.

Adegoke, A. O., Oyeleke, O. D., Mahmud, B., Ajoje, J. O., & Thomase, S. (2019). Design and Construction of an Obstacle-Detecting Glasses for the Visually Impaired, (July), 57–66. doi:10.5815/ijem.2019.04.05

Afif, M., Ayachi, R., Said, Y., Pissaloux, E., & Atri, M. (2020). An evaluation of retinanet on indoor object detection for blind and visually impaired persons assistance navigation. *Neural Processing Letters*, 1–15.

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Agarwal, R., Frosst, N., Zhang, X., Caruana, R., & Hinton, G. E. (2020). Neural additive models: Interpretable machine learning with neural nets. *arXiv preprint arXiv:2004.13912*.

Ahmed, H., Ullah, I., Khan, U., Qureshi, M. B., Manzoor, S., Muhammad, N., Khan, S., et al. (2019). Adaptive Filtering on GPS-Aided MEMS-IMU for Optimal Estimation of Ground Vehicle Trajectory. *Sensors*, *19*(24), 5357.

Alhashim, I., & Wonka, P. (2018). High Quality Monocular Depth Estimation via Transfer Learning. *arXiv preprint arXiv:1812.11941*.

Amin-Naji, M., Aghagolzadeh, A., & Ezoji, M. (2019). Ensemble of CNN for multi-focus image fusion. *Information fusion*, *51*, 201–214.

Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I., & Atkinson, P. M. (2021). Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *11*(5), e1424.

Anwar, S. M., Majid, M., Qayyum, A., Awais, M., Alnowami, M., & Khan, M. K. (2018). Medical image analysis using convolutional neural networks: a review. *Journal of medical systems*, *42*(11), 1–13.

Aqel, M. O., Marhaban, M. H., Saripan, M. I., & Ismail, N. B. (2016). Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, *5*(1), 1897.

Arauzo-Azofra, A., Aznarte, J. L., & Benıtez, J. M. (2011). Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert systems with applications*, *38*(7), 8170–8177.

Arebi, N., Swain, D., Suzuki, N., Fraser, C., Price, A., & Saunders, B. P. (2007). Endoscopic mucosal resection of 161 cases of large sessile or flat colorectal polyps. *Scandinavian journal of gastroenterology*, *42*(7), 859–866.

Arrieta, A. B., Dıaz-Rodrıguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcıa, S., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82–115.

Bai, J., Liu, Z., Lin, Y., Li, Y., Lian, S., & Liu, D. (2019). Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People. *Electronics*, *8*(6), 697. doi:10.3390/electronics8060697

Balamurugan, G., Valarmathi, J., & Naidu, V. (2016). Survey on UAV navigation in GPS denied environments. *2016 International conference on signal processing, communication, power and embedded system (SCOPES)* (pp. 198–204). IEEE.

Balashova, E., Wang, J., Singh, V., Georgescu, B., Teixeira, B., & Kapoor, A. (2019). 3D Organ Shape Reconstruction from Topogram Images. *International Conference on Information Processing in Medical Imaging* (pp. 347–359). Springer.

Baldi, P., & Sadowski, P. (2014). The dropout learning algorithm. *Artificial intelligence*, *210*, 78–122.

Ballard, D. H. (1987). Modular learning in neural networks. *Aaai* (Vol. 647, pp. 279–284).

Barbiero, P., Ciravegna, G., Giannini, F., Lió, P., Gori, M., & Melacci, S. (2022). Entropy-based logic explanations of neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, pp. 6046–6054).

Barlow, H. B. (1989). Unsupervised learning. *Neural computation*, *1*(3), 295–311.

Barontini, F., Catalano, M. G., Pallottino, L., Leporini, B., & Bianchi, M. (2020). Integrating wearable haptics and obstacle avoidance for the visually impaired in indoor navigation: A user-centered approach. *IEEE Transactions on Haptics*, *14*(1), 109–122.

Bashiri, F. S., LaRose, E., Badger, J. C., D'Souza, R. M., Yu, Z., & Peissig, P. (2018). Object Detection to Assist Visually Impaired People: A Deep Neural Network Adventure (Vol. 1, pp. 500–510). Springer International Publishing. doi:10.1007/978-3-030-03801-4_44

Basodi, S., Ji, C., Zhang, H., & Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, *3*(3), 196–207.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6541–6549).

Bauer, Z., Dominguez, A., Cruz, E., Gomez-Donoso, F., Orts-Escolano, S., & Cazorla, M. (2020). Enhancing perception for the visually impaired with deep learning techniques and low-cost wearable sensors. *Pattern recognition letters*, *137*, 27–36.

Ben-Shabat, Y., Koneputugodage, C. H., & Gould, S. (2022). DiGS: Divergence guided shape implicit neural representation for unoriented point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 19323–19332).

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bernal, J., Sánchez, F. J., Vilarino, F., Arnold, M., Ghosh, A., & Lacey, G. (2014). Experts vs. novices: applying eye-tracking methodologies in colonoscopy video screening for polyp search. *Proceedings of the Symposium on Eye Tracking Research and Applications* (pp. 223–226). ACM.

Bernard, S., & Pike, S. (2015). Isotopic analysis of marble from the Stoa of Attalos in the Athenian Agora and the Hellenistic quarries of Mount Pentelikon. *Isotopic analysis of marble from the Stoa of Attalos in the Athenian Agora and the Hellenistic quarries of Mount Pentelikon*, 451–459.

Biederman, I., & Ju, G. (1988). Surface versus edge-based determinants of visual recognition. *Cognitive psychology*, *20*(1), 38–64.

Borji, A., Cheng, M.-M., Jiang, H., & Li, J. (2015). Salient object detection: A benchmark. *IEEE transactions on image processing*, *24*(12), 5706–5722.

Bottega, G. H., & Balbinot, A. (2020). Proposal of an obstacle detector with sound response for the visually impaired. *Health and Technology*, *10*(3), 739–757.

Bouguet, J.-Y. (2004). Camera calibration toolbox for matlab. *http://www. vision. caltech. edu/bouguetj/calib_doc/index. html*.

Brassai, S. T., Iantovics, B., & Enachescu, C. (2012). Optimization of robotic mobile agent navigation. *Studies in Informatics and Control*, *21*(4), 403–412.

Bylinskii, Z., Recasens, A., Borji, A., Oliva, A., Torralba, A., & Durand, F. (2016). Where should saliency models look next? *European Conference on Computer Vision* (pp. 809–824). Springer.

Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2022). Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 1–12.

Cai, Y., Zheng, J., Zhang, X., Jiang, H., & Huang, M.-C. (2020). GAM feature selection to discover predominant factors for mortality of weekend and weekday admission to the ICUs. *Smart Health*, *18*, 100145.

Caraiman, S., Morar, A., Owczarek, M., Burlacu, A., Rzeszotarski, D., Botezatu, N., Herghelegiu, P., et al. (2017). Computer Vision for the Visually Impaired: The Sound of Vision System. *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on* (pp. 1480–1489). IEEE.

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, *538*(7623), 20.

Cawley, G., Talbot, N., & Girolami, M. (2006). Sparse multinomial logistic regression via bayesian l1 regularisation. *Advances in neural information processing systems*, *19*.

Chabra, R., Lenssen, J. E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., & Newcombe, R. (2020). Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. *European Conference on Computer Vision* (pp. 608–625). Springer.

Chaney, K., Bucher, B., Chatzipantazis, E., Shi, J., & Daniilidis, K. (n.d.). Unsupervised Monocular Depth and Latent Structure.

Chatterjee, S., & Callaway, E. M. (2003). Parallel colour-opponent pathways to primary visual cortex. *Nature*, *426*(6967), 668–671.

Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., & Su, J. K. (2019). This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, *32*.

Chen, H., & Li, Y. (2018). Progressively complementarity-aware fusion network for RGB-D salient object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3051–3060).

Chen, H., & Li, Y. (2019). Three-stream attention-aware network for RGB-D salient object detection. *IEEE Transactions on Image Processing*, *28*(6), 2825–2835.

Chen, H., Li, Y., & Su, D. (2019). Multi-modal fusion network with multi-scale multi-path and cross-modal interactions for RGB-D salient object detection. *Pattern Recognition*, *86*, 376–385.

Chen, H., Zhang, Y., Yang, K., Martinez, M., Müller, K., & Stiefelhagen, R. (2020). Can we unify perception and localization in assisted navigation? An indoor semantic visual positioning

system for visually impaired people. *International Conference on Computers Helping People with Special Needs* (pp. 97–104). Springer.

Chen, L., Bentley, P., Mori, K., Misawa, K., Fujiwara, M., & Rueckert, D. (2019). Self-supervised learning for medical image analysis using image context restoration. *Medical image analysis*, *58*, 101539.

Chen, L., Guo, B., & Sun, W. (2010). Obstacle detection system for visually impaired people based on stereo vision. *2010 Fourth International Conference on Genetic and Evolutionary Computing* (pp. 723–726). IEEE.

Chen, R., Chen, H., Ren, J., Huang, G., & Zhang, Q. (2019). Explaining neural networks semantically and quantitatively. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9187–9196).

Chen, S., Yao, D., Cao, H., & Shen, C. (2019). A Novel Approach to Wearable Image Recognition Systems to Aid Visually Impaired People. *Applied Sciences*, *9*(16), 3350. doi:10.3390/app9163350

Chen, Y., Wang, Y., Gu, Y., He, X., Ghamisi, P., & Jia, X. (2019). Deep learning ensemble for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *12*(6), 1882–1897.

Chen, Z., & Zhang, H. (2019). Learning implicit fields for generative shape modeling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5939–5948).

Cheng, R., Wang, K., Bai, J., & Xu, Z. (2019). OpenMPR: Recognize places using multimodal data for people with visual impairments. *Measurement Science and Technology*, *30*(12), 124004. doi:10.1088/1361-6501/ab2106

Chibane, J., Pons-Moll, G., & others. (2020). Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, *33*, 21638–21652.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., & others. (2008). Meshlab: an open-source mesh processing tool. *Eurographics Italian chapter conference* (Vol. 2008, pp. 129–136). Salerno, Italy.

Ciuti, G., Caliò, R., Camboni, D., Neri, L., Bianchi, F., Arezzo, A., Koulaouzidis, A., et al. (2016). Frontiers of robotic endoscopic capsules: a review. *Journal of micro-bio robotics*, *11*(1), 1–18.

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., et al. (2013). The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository. *Journal of digital imaging*, *26*(6), 1045–1057.

Classical Studies, A. S. of. (n.d.). https://www.ascsa.net. Retrieved from https://www.ascsa.net

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Cornia, M., Baraldi, L., Serra, G., & Cucchiara, R. (2018). Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing*, *27*(10), 5142–5154.

Covert, I., Lundberg, S., & Lee, S.-I. (2020). Understanding Global Feature Contributions With Additive Importance Measures.

Craye, C., Filliat, D., & Goudou, J.-F. (2016). Environment exploration for object-based visual saliency learning. *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2303–2309). IEEE.

Criminisi, A., Reid, I., & Zisserman, A. (2000). Single view metrology. *International Journal of Computer Vision*, *40*(2), 123–148.

D.K. Iakovidis, E. S. D. Diamantis G. Dimas C. Ntakolia. (2019). Improved Mobility for the Visually Impaired. In S. P. (eds) (Ed.), . Springer.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255). Ieee.

Deng, X., Xiang, Y., Mousavian, A., Eppner, C., Bretl, T., & Fox, D. (2020). Self-supervised 6d object pose estimation for robot manipulation. *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3665–3671). IEEE.

Deng, Z., Yao, Y., Deng, B., & Zhang, J. (2021). A robust loss for point cloud registration. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 6138–6147).

Deroy, O. (2013). Object-sensitivity versus cognitive penetrability of perception. *Philosophical studies*, *162*(1), 87–107.

Diamantis, D. E., & Iakovidis, D. K. (2020). Fuzzy pooling. *IEEE Transactions on Fuzzy Systems*, *29*(11), 3481–3488.

Diamantis, D. E., Iakovidis, D. K., & Koulaouzidis, A. (2019). Look-behind fully convolutional neural network for computer-aided endoscopy. *Biomedical signal processing and control*, *49*, 192–201.

Dijk, T. van. (2020). Self-Supervised Learning for Visual Obstacle Avoidance.

Dimas, G., Bianchi, F., Iakovidis, D. K., Karargyris, A., Ciuti, G., & Koulaouzidis, A. (2020). Endoscopic single-image size measurements. *Measurement Science and Technology*, *31*(7), 074010.

Dimas, G., Cholopoulou, E., & Iakovidis, D. K. (2021). Self-Supervised Soft Obstacle Detection for Safe Navigation of Visually Impaired People. *2021 IEEE International Conference on Imaging Systems and Techniques (IST)* (pp. 1–6). IEEE.

Dimas, G., Cholopoulou, E., & Iakovidis, D. K. (2022). E Pluribus Unum Interpretable Convolutional Neural Networks. *arXiv preprint arXiv:2208.05369*.

Dimas, G., Diamantis, D. E., Kalozoumis, P., & Iakovidis, D. K. (2020). Uncertainty-Aware Visual Perception System for Outdoor Navigation of the Visually Challenged. *Sensors*, *20*(8), 2385.

Dimas, G., Gatoula, P., & Iakovidis, D. K. (2021). MonoSOD: Monocular Salient Object Detection based on Predicted Depth. *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4377–4383). IEEE.

Dimas, G., Iakovidis, D. K., Karargyris, A., Ciuti, G., & Anastasios, K. (2017). An artificial neural network architecture for non-parametric visual odometry in wireless capsule endoscopy. *Measurement Science and Technology*, *28*(9), 94005. Retrieved from http://stacks.iop.org/0957-0233/28/i=9/a=094005

Dimas, G., Iakovidis, D., & Koulaouzidis, A. (2019). MedGaze: Gaze Estimation on WCE Images Based on a CNN Autoencoder. *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)* (pp. 363–367). IEEE.

Dimas, G., Iakovidis, D., & Koulaouzidis, A. (2019). MedGaze: Gaze Estimation on WCE Images Based on a CNN Autoencoder. *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)* (pp. 363–367). doi:10.1109/BIBE.2019.00071

Dimas, G., Ntakolia, C., & Iakovidis, D. K. (2019). Obstacle detection based on generative adversarial networks and fuzzy sets for computer-assisted navigation. *International Conference on Engineering Applications of Neural Networks* (pp. 533–544). Springer.

Dimas, G., Spyrou, E., Iakovidis, D. K., & Koulaouzidis, A. (2017a). Intelligent visual localization of wireless capsule endoscopes enhanced by color information. *Computers in biology and medicine*, *89*, 429–440.

Dimas, G., Spyrou, E., Iakovidis, D. K., & Koulaouzidis, A. (2017b). Intelligent visual localization of wireless capsule endoscopes enhanced by color information. *Computers in Biology and Medicine*, *89*, 429–440. doi:10.1016/j.compbiomed.2017.08.029

Doraisamy, S., Golzari, S., Mohd, N., Sulaiman, M. N., & Udzir, N. I. (2008). A Study on Feature Selection and Classification Techniques for Automatic Genre Classification of Traditional Malay Music. *ISMIR* (pp. 331–336).

Dozat, T. (2016). Incorporating nesterov momentum into adam.

Duch, W., Winiarski, T., Biesiada, J., & Kachel, A. (2003). Feature selection and ranking filters. *International conference on artificial neural networks (ICANN) and International conference on neural information processing (ICONIP)* (Vol. 251, p. 254).

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

Duh, P.-J., Sung, Y.-C., Chiang, L.-Y. F., Chang, Y.-J., & Chen, K.-W. (2020). V-eye: A vision-based navigation system for the visually impaired. *IEEE Transactions on Multimedia*, *23*, 1567–1580.

Fan, D.-P., Cheng, M.-M., Liu, Y., Li, T., & Borji, A. (2017). Structure-measure: A new way to evaluate foreground maps. *Proceedings of the IEEE international conference on computer vision* (pp. 4548–4557).

Fan, D.-P., Gong, C., Cao, Y., Ren, B., Cheng, M.-M., & Borji, A. (2018). Enhanced-alignment measure for binary foreground map evaluation. *arXiv preprint arXiv:1805.10421*.

Fan, D.-P., Lin, Z., Zhang, Z., Zhu, M., & Cheng, M.-M. (2020). Rethinking RGB-D Salient Object Detection: Models, Data Sets, and Large-Scale Benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*.

Feferman, S., Dawson, J. W., Kleene, S. C., Moore, G. H., & Solovay, R. M. (1998). Kurt Gödel: Collected Works, Vol. I: Publications 1929-1936.

Fox, S. E., & Faulkner-Jones, B. E. (2017). Eye-Tracking in the Study of Visual Expertise: Methodology and Approaches in Medicine. *Frontline Learning Research*, *5*(3), 29–40.

Fradkov, A. L. (2020). Early history of machine learning. *IFAC-PapersOnLine*, *53*(2), 1385–1390.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. *Competition and cooperation in neural nets* (pp. 267–285). Springer.

Gao, F., & Han, L. (2012). Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, *51*(1), 259–277.

Gao, M., Jiang, J., Zou, G., John, V., & Liu, Z. (2019). RGB-D-based object recognition using multimodal convolutional neural networks: A survey. *IEEE Access*, *7*, 43110–43136.

Gatoula, P., Dimas, G., Iakovidis, D. K., & Koulaouzidis, A. (2021). Enhanced CNN-Based Gaze Estimation on Wireless Capsule Endoscopy Images. *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)* (pp. 189–195). IEEE.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE conference on computer vision and pattern recognition* (pp. 3354–3361). IEEE.

Ghahramani, Z. (2003). Unsupervised learning. *Summer school on machine learning* (pp. 72–112). Springer.

Godard, C., Mac Aodha, O., Firman, M., & Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. *Proceedings of the IEEE international conference on computer vision* (pp. 3828–3838).

Goldstein, O., Segol, O., Gross, S. A., Jacob, H., & Siersema, P. D. (2017). Novel device for measuring polyp size: an ex vivo animal study. *Gut*, *67*(10), 1755–1756. doi:10.1136/gutjnl-2017-314829

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., et al. (2014). Generative adversarial nets. *Advances in neural information processing systems* (pp. 2672–2680).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., et al. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Greisdorf, H., & O'Connor, B. (2002). Modelling what users see when they look at images: a cognitive viewpoint. *Journal of documentation*.

Gropp, A., Yariv, L., Haim, N., Atzmon, M., & Lipman, Y. (2020). Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*.

Grunnet-Jepsen, A., Sweetser, J. N., Winer, P., Takagi, A., & Woodfill, J. (2018). Projectors for intel® realsense™ depth cameras d4xx. *Intel Support, Interl Corporation: Santa Clara, CA, USA*.

Grunnet-Jepsen, A., Sweetser, J. N., & Woodfill, J. (2018). Best-Known-Methods for Tuning Intel® RealSense™ D400 Depth Cameras for Best Performance. *Intel Corporation: Satan Clara, CA, USA*, *1*.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., et al. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, *77*, 354–377.

Ha, D., Dai, A., & Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Han, J., Chen, H., Liu, N., Yan, C., & Li, X. (2017). CNNs-based RGB-D saliency detection via cross-view transfer and multiview fusion. *IEEE transactions on cybernetics*, *48*(11), 3171–3183.

Hansen, T., & Gegenfurtner, K. R. (2009). Independence of color and luminance edges in natural scenes. *Visual neuroscience*, *26*(1), 35–49.

Harel, J., Koch, C., & Perona, P. (2007). Graph-based visual saliency.

Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Hastie, T. J., & Tibshirani, R. J. (1990). *Generalized additive models* (Vol. 43). CRC press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Heikkila, J., & Silvén, O. (1997). A four-step camera calibration procedure with implicit image correction. *Proceedings of IEEE computer society conference on computer vision and pattern recognition* (pp. 1106–1112). IEEE.

Heinrich, S. (2002). Fast obstacle detection using flow/depth constraint. *Intelligent Vehicle Symposium, 2002. IEEE* (Vol. 2, pp. 658–665). IEEE.

Hengle, A., Kulkarni, A., Bavadekar, N., Kulkarni, N., & Udyawar, R. (2020). Smart Cap: A Deep Learning and IoT Based Assistant for the Visually Impaired. *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 1109–1116). IEEE.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hoiem, D., Efros, A. A., & Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, *80*(1), 3–15.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Hsieh, Y.-Z., Lin, S.-S., & Xu, F.-X. (2020). Development of a wearable guide device based on convolutional neural network for blind or visually impaired persons. *Multimedia Tools and Applications*, *79*(39), 29473–29491.

Hu, M., Penney, G., Edwards, P., Figl, M., & Hawkes, D. J. (2007). 3D reconstruction of internal organ surfaces for minimal invasive surgery. *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 68–77). Springer.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700–4708).

Huang, H., He, R., Sun, Z., & Tan, T. (2017). Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1689–1697).

Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., Han, X., et al. (2020). Unet 3+: A full-scale connected unet for medical image segmentation. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1055–1059). IEEE.

Huang, L., Li, G., Li, Y., & Lin, L. (2019). Lightweight Contrast Modeling for Attention-Aware Visual Localization. *2019 International Conference on Robotics and Automation (ICRA)* (pp. 2104–2110). IEEE.

Huang, P.-W., & Dai, S. (2003). Image retrieval by texture similarity. *Pattern recognition*, *36*(3), 665–679.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106.

Huovilainen, A. (2004). Non-linear digital implementation of the Moog ladder filter. *Proceedings of the International Conference on Digital Audio Effects (DAFx-04)* (pp. 61–64).

Hurvich, L. M., & Jameson, D. (1957). An opponent-process theory of color vision. *Psychological review*, *64*(6p1), 384.

Hussmann, S., Ringbeck, T., & Hagebeuker, B. (2008). A performance review of 3D TOF vision systems in comparison to stereo vision systems. *Stereo vision*, *372*.

Iakovidis, D. K., Diamantis, D., Dimas, G., Ntakolia, C., & Spyrou, E. (2020). Digital enhancement of cultural experience and accessibility for the visually impaired. *Technological Trends in Improved Mobility of the Visually Impaired* (pp. 237–271). Springer.

Iakovidis, D. K., Dimas, G., Karargyris, A., Bianchi, F., Ciuti, G., & Koulaouzidis, A. (2018). Deep endoscopic visual measurements. *IEEE Journal of Biomedical and Health Informatics*, *23*(6), 2211–2219.

Iakovidis, D. K., Dimas, G., Karargyris, A., Bianchi, F., Ciuti, G., & Koulaouzidis, A. (2019). Deep Endoscopic Visual Measurements. *{IEEE} Journal of Biomedical and Health Informatics*, *23*(6), 2211–2219. doi:10.1109/jbhi.2018.2853987

Iakovidis, D. K., Georgakopoulos, S. V., Vasilakakis, M., Koulaouzidis, A., & Plagianakos, V. P. (2018). Detecting and locating gastrointestinal anomalies using deep learning and iterative cluster unification. *IEEE transactions on medical imaging*, *37*(10), 2196–2210.

Iakovidis, D. K., & Koulaouzidis, A. (2014). Automatic lesion detection in wireless capsule endoscopy—a simple solution for a complex problem. *2014 IEEE International Conference on Image Processing (ICIP)* (pp. 2236–2240). IEEE.

Iakovidis, D. K., & Koulaouzidis, A. (2015). Software for enhanced video capsule endoscopy: challenges for essential progress. *Nature Reviews Gastroenterology & Hepatology*, *12*(3), 172.

Iakovidis, D. K., Ooi, M., Kuang, Y. C., Demidenko, S., Shestakov, A., Sinitsin, V., Henry, M., et al. (2021). Roadmap on signal processing for next generation measurement systems. *Measurement Science and Technology*, *33*(1), 012002.

Iakovidis, D. K., Tsevas, S., & Polydorou, A. (2010). Reduction of capsule endoscopy reading times by unsupervised image mining. *Computerized Medical Imaging and Graphics*, *34*(6), 471–478.

Ishikawa, R., Hachiuma, R., & Saito, H. (2021). Self-Supervised Audio-Visual Feature Learning for Single-Modal Incremental Terrain Type Clustering. *IEEE Access*, *9*, 64346–64357.

Islam, M. M., Sadi, M. S., & Bräunl, T. (2020). Automated walking guide to enhance the mobility of visually impaired people. *IEEE Transactions on Medical Robotics and Bionics*, *2*(3), 485–496.

Islam, M. T., Ahmad, M., & Bappy, A. S. (2020). Microprocessor-Based Smart Blind Glass System for Visually Impaired People. *Studies in Computational Intelligence* (Vol. 669, pp. 151–161). doi:10.1007/978-981-13-7564-4_13

Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New phytologist*, *11*(2), 37–50.

Jampani, V., Sivaswamy, J., Vaidya, V., & others. (2012). Assessment of computational visual attention models on medical images. *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing* (p. 80). ACM.

Jia, S., & Bruce, N. D. (2018). Eml-net: An expandable multi-layer network for saliency prediction. *arXiv preprint arXiv:1805.01047*.

Jiang, B., Yang, J., Lv, Z., & Song, H. (2019). Wearable vision assistance system based on binocular sensors for visually impaired users. *IEEE Internet of Things Journal*, *6*(2), 1375–1383. doi:10.1109/JIOT.2018.2842229

Jiang, M., Huang, S., Duan, J., & Zhao, Q. (2015). Salicon: Saliency in context. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1072–1080).

Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. *European conference on computer vision* (pp. 694–711). Springer.

Joshi, R. C., Yadav, S., Dutta, M. K., & Travieso-Gonzalez, C. M. (2020). Efficient Multi-Object Detection and Smart Navigation Using Artificial Intelligence for Visually Impaired People. *Entropy*, *22*(9), 941.

Ju, R., Ge, L., Geng, W., Ren, T., & Wu, G. (2014). Depth saliency based on anisotropic center-surround difference. *2014 IEEE international conference on image processing (ICIP)* (pp. 1115–1119). IEEE.

Judd, T., Ehinger, K., Durand, F., & Torralba, A. (2009). Learning to predict where humans look. *2009 IEEE 12th international conference on computer vision* (pp. 2106–2113). IEEE.

Jung, J. H., & Kwon, Y. (2021). Color, edge, and pixel-wise explanation of predictions based on interpretable neural network model. *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 6003–6010). IEEE.

Just, M. A., & Carpenter, P. A. (1984). Using eye fixations to study reading comprehension. *New methods in reading comprehension research*, 151–182.

Kahn, G., Abbeel, P., & Levine, S. (2021). Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, *6*(2), 1312–1319.

Kalozoumis, P., Dimas, G., Triantafyllou, G., & Iakovidis, D. K. (2022). A Framework for the Development of Intestinal Digital Twins Integrating Machine Learning and Multiphysics Modelling. *Virtual Physiological Human Conference*.

Kalozoumis, P. G., Marino, M., Carniel, E. L., & Iakovidis, D. K. (2022). Towards the Development of a Digital Twin for Endoscopic Medical Device Testing. *Digital Twins for Digital Transformation: Innovation in Industry*. Springer.

Kapishnikov, A., Bolukbasi, T., Viégas, F., & Terry, M. (2019). Xrai: Better attributions through regions. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 4948–4957).

Kaur, B., & Bhattacharya, J. (2018). A scene perception system for visually impaired based on object detection and classification using multi-modal DCNN. *arXiv preprint arXiv:1805.08798*.

Kaz, A. M., Anwar, A., O'Neill, D. R., & Dominitz, J. A. (2016). Use of a novel polyp "ruler snare" improves estimation of colon polyp size. *Gastrointestinal endoscopy*, *83*(4), 812–816.

Khan, S. N., Muhammad, N., Farwa, S., Saba, T., Khattak, S., & Mahmood, Z. (2019). Early Cu depth decision and reference picture selection for low complexity Mv-Hevc. *Symmetry*, *11*(4), 454.

Kim, J., & Canny, J. (2017). Interpretable learning for self-driving cars by visualizing causal attention. *Proceedings of the IEEE international conference on computer vision* (pp. 2942–2950).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klein, D. A., Illing, B., Gaspers, B., Schulz, D., & Cremers, A. B. (2017). Hierarchical salient object detection for assisted grasping. *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2230–2237). IEEE.

Kondratyuk, D., Tan, M., Brown, M. A., & Gong, B. (2020). When Ensembling Smaller Models is More Efficient than Single Large Models. *ArXiv*, *abs/2005.00570*.

Kotoulas, L., & Andreadis, I. (2005). Image analysis using moments. *5th Int. Conf. on Technology and Automation, Thessaloniki, Greece* (Vol. 360364).

Koulaouzidis, A., Iakovidis, D. K., Yung, D. E., Rondonotti, E., Kopylov, U., Plevris, J. N., Toth, E., et al. (2017). KID Project: an internet-based digital video atlas of capsule endoscopy for research purposes. *Endosc Int Open*, *5*(6), E477–E483.

Koulaouzidis, A., Iakovidis, D. K., Yung, D. E., Rondonotti, E., Kopylov, U., Plevris, J. N., Toth, E., et al. (2017). KID Project: an internet-based digital video atlas of capsule endoscopy for research purposes. *Endoscopy international open*, *5*(6), E477.

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2176–2184).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Kubat, M., & Kubat. (2017). *An introduction to machine learning* (Vol. 2). Springer.

Kume, K., Watanabe, T., Yoshikawa, I., & Harada, M. (2014). Endoscopic Measurement of Polyp Size Using a Novel Calibrated Hood. *Gastroenterology Research and Practice*, *2014*, 1–4. doi:10.1155/2014/714294

Kummerer, M., Wallis, T. S., Gatys, L. A., & Bethge, M. (2017). Understanding low-and high-level contributions to fixation prediction. *Proceedings of the IEEE International Conference on Computer Vision* (pp. 4789–4798).

Kurobe, A., Nakajima, Y., Saito, H., & Kitani, K. (2020). Audio-visual self-supervised terrain type discovery for mobile platforms. *arXiv preprint arXiv:2010.06318*.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1675–1684).

Lazzarini, V., & Timoney, J. (2010). New perspectives on distortion synthesis for virtual analog oscillators. *Computer Music Journal*, *34*(1), 28–40.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, C.-H., Su, Y.-C., & Chen, L.-G. (2012). An intelligent depth-based obstacle detection system for visually-impaired aid applications. *2012 13th International Workshop on Image Analysis for Multimedia Interactive Services* (pp. 1–4). IEEE.

Lehr, D., & Ohm, P. (2017). Playing with the data: what legal scholars should learn about machine learning. *UCDL Rev.*, *51*, 653.

Lévêque, L., Bosmans, H., Cockmartin, L., & Liu, H. (2018). State of the art: Eye-tracking studies in medical imaging. *IEEE Access*, *6*, 37023–37034.

Lévêque, L., Zhang, W., Parker, P., & Liu, H. (2017). The impact of specialty settings on the perceived quality of medical ultrasound video. *IEEE Access*, *5*, 16998–17005.

Levina, E., & Bickel, P. (2001). The earth mover's distance is the mallows distance: Some insights from statistics. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* (Vol. 2, pp. 251–256). IEEE.

Lewiner, T., Lopes, H., Vieira, A. W., & Tavares, G. (2003). Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, *8*(2), 1–15.

Li, W., Duan, L., Xu, D., & Tsang, I. W. (2013). Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, *36*(6), 1134–1148.

Li, Y., & Wang, Z. (2020). RGB line pattern-based stereo vision matching for single-shot 3-D measurement. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–13.

Liang, H., Ouyang, Z., Zeng, Y., Su, H., He, Z., Xia, S.-T., Zhu, J., et al. (2020). Training interpretable convolutional neural networks by differentiating class-specific filters. *European Conference on Computer Vision* (pp. 622–638). Springer.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

Liu, K., Zhang, M., & Pan, Z. (2016). Facial expression recognition with CNN ensemble. *2016 international conference on cyberworlds (CW)* (pp. 163–166). IEEE.

Liu, N., & Han, J. (2016). Dhsnet: Deep hierarchical saliency network for salient object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 678–686).

Liu, N., & Han, J. (2018). A deep spatial contextual long-term recurrent convolutional network for saliency detection. *IEEE Transactions on Image Processing*, *27*(7), 3264–3274.

Liu, O.-C. A., Li, S.-K., Yan, L.-Q., Ng, S.-C., & Kwok, C.-P. (2020). A Visually Impaired Assistant Using Neural Network and Image Recognition with Physical Navigation. *International Symposium on Neural Networks* (pp. 270–281). Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *European conference on computer vision* (pp. 21–37). Springer.

Liu, Y.-C., Hsieh, Y.-A., Chen, M.-H., Yang, C.-H. H., Tegner, J., & Tsai, Y.-C. J. (2020). Interpretable self-attention temporal reasoning for driving behavior understanding. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2338–2342). IEEE.

Long, N., Wang, K., Cheng, R., Hu, W., & Yang, K. (2019). Unifying obstacle detection, recognition, and fusion based on millimeter wave radar and RGB-depth sensors for the visually impaired. *Review of Scientific Instruments*, *90*(4). doi:10.1063/1.5093279

Lu, M. Y., Chen, R. J., Wang, J., Dillon, D., & Mahmood, F. (2019). Semi-supervised histology classification using deep multiple instance learning and contrastive predictive coding. *arXiv preprint arXiv:1910.10825*.

Lukac, R., Plataniotis, K. N., & Hatzinakos, D. (2005). Color image zooming on the Bayer pattern. *IEEE Transactions on Circuits and Systems for Video Technology*, *15*(11), 1475–1492.

Lundberg, S. M., & Lee, S.-I. (2017a). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*.

Lundberg, S. M., & Lee, S.-I. (2017b). A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 4765–4774). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Ma, B., Han, Z., Liu, Y.-S., & Zwicker, M. (2020). Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. *arXiv preprint arXiv:2011.13495*.

Mäenpää, T., & Pietikäinen, M. (2004). Classification with color and texture: jointly or separately? *Pattern recognition*, *37*(8), 1629–1640.

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*, 381–386.

Mahmood, F., Chen, R., Sudarsky, S., Yu, D., & Durr, N. J. (2018). Deep learning with cinematic rendering: fine-tuning deep neural networks using photorealistic medical images. *Physics in Medicine & Biology*, *63*(18), 185012.

Mahmood, Z., Bibi, N., Usman, M., Khan, U., & Muhammad, N. (2019). Mobile cloud based-framework for sports applications. *Multidimensional Systems and Signal Processing*, *30*(4), 1991–2019.

Mallat, S. G. (2009). A theory for multiresolution signal decomposition: the wavelet representation. *Fundamental Papers in Wavelet Theory* (pp. 494–513). Princeton University Press.

Mancini, M., Costante, G., Valigi, P., & Ciarfuglia, T. A. (2018). J-MOD 2: Joint Monocular Obstacle Detection and Depth Estimation. *IEEE Robotics and Automation Letters*, *3*(3), 1490–1497.

Martinez, M., Yang, K., Constantinescu, A., & Stiefelhagen, R. (2020). Helping the blind to get through covid-19: Social distancing assistant using real-time semantic segmentation on rgb-d video. *Sensors*, *20*(18), 5202.

Mendi, E., & Milanova, M. (2009). Image segmentation with active contours based on selective visual attention. *WSEAS International Conference. Proceedings. Mathematics and*

Computers in Science and Engineering. World Scientific and Engineering Academy and Society.

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., & Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4460–4470).

Michel, O. (2004). Webots: Professional Mobile Robot Simulation. *Journal of Advanced Robotics Systems*, *1*(1), 39–42. Retrieved from http://www.ars-journal.com/International-Journal-of- Advanced-Robotic-Systems/Volume-1/39-42.pdf

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. *European conference on computer vision* (pp. 405–421). Springer.

Ming, H., & Xu, K. (2019, August). Surface Blemishes of Aluminum Material Image Recognition Based on Transfer Learning. In Journal of Physics: Conference Series (Vol. 1288, No. 1, p. 012016). IOP Publishing.

Minsky, M., & Papert, S. A. (2017). *Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry*. MIT press.

Mitsou, A., Koutsiou, D.-C. C., Diamantis, D. E., Psallidas, T., Dimas, G., Vasilakakis, M., Kalozoumis, P., et al. (2022). ENORASI Assistive Computer Vision-based System for the Visually Impaired: A User Evaluation Study. *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments* (pp. 668–677).

Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in AI. *Proceedings of the conference on fairness, accountability, and transparency* (pp. 279–288).

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Mozaffari, S., Al-Jarrah, O. Y., Dianati, M., Jennings, P., & Mouzakitis, A. (2020). Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. *IEEE Transactions on Intelligent Transportation Systems*, *23*(1), 33–47.

Muhammad, K., Ahmad, J., Sajjad, M., & Baik, S. W. (2016). Visual saliency models for summarization of diagnostic hysteroscopy videos in healthcare systems. *SpringerPlus*, *5*(1), 1495.

Muñoz-Romero, S., Gorostiaga, A., Soguero-Ruiz, C., Mora-Jiménez, I., & Rojo-Álvarez, J. L. (2020). Informative variable identifier: Expanding interpretability in feature selection. *Pattern Recognition*, *98*, 107077.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, *116*(44), 22071–22080.

Nachbar, F., Stolz, W., Merkle, T., Cognetta, A. B., Vogt, T., Landthaler, M., Bilek, P., et al. (1994). The ABCD rule of dermatoscopy: high prospective value in the diagnosis of doubtful melanocytic skin lesions. *Journal of the American Academy of Dermatology*, *30*(4), 551–559.

Nam, J., & Kim, S. (2015). Heterogeneous defect prediction. *Proceedings of the 2015 10th joint meeting on foundations of software engineering* (pp. 508–519).

Navab, N., Hornegger, J., Wells, W. M., & Frangi, A. (2015). *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* (Vol. 9351). Springer.

Ndou, N., Ajoodha, R., & Jadhav, A. (2021). Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches. *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)* (pp. 1–6). IEEE.

Neela Maadhuree, A., Mathews, R. S., & Rene Robin, C. R. (2020). Le Vision: An Assistive Wearable Device for the Visually Challenged. *Advances in Intelligent Systems and Computing* (Vol. 207 AISC, pp. 353–361). doi:10.1007/978-3-030-16660-1_35

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence O (1/k 2). *Doklady an ussr* (Vol. 269, pp. 543–547).

Nguyen, H. T., Walker, C., & Walker, E. A. (2018). *A first course in fuzzy logic*. Chapman and Hall/CRC.

Niu, Y., Geng, Y., Li, X., & Liu, F. (2012). Leveraging stereopsis for saliency analysis. *2012 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 454–461). IEEE.

Novikoff, A. B. (1963). *On convergence proofs for perceptrons*. STANFORD RESEARCH INST MENLO PARK CA.

Ntakolia, C., Dimas, G., & Iakovidis, D. K. (2020). User-centered system design for assisted navigation of visually impaired individuals in outdoor cultural environments. *Universal Access in the Information Society*, 1–26.

Oh, J., & Han, J. (2021). Pedestrian travel distance estimation using optical flow and smartphone sensors. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–8.

Oka, K., Seki, T., Akatsu, T., Wakabayashi, T., Inui, K., & Yoshino, J. (2014). Clinical study using novel endoscopic system for measuring size of gastrointestinal lesion. *World Journal of Gastroenterology: WJG*, *20*(14), 4050.

Osher, S., & Fedkiw, R. (2003). Signed distance functions. *Level set methods and dynamic implicit surfaces* (pp. 17–22). Springer.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, *32*(2), 604–624.

Pakarinen, J., & Yeh, D. T. (2009). A review of digital techniques for modeling vacuum-tube guitar amplifiers. *Computer Music Journal*, *33*(2), 85–100.

Pan, J., Ferrer, C. C., McGuinness, K., O'Connor, N. E., Torres, J., Sayrol, E., & Giro-i-Nieto, X. (2017). Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081*.

Pan, S., & Yang, Q. (2010). A survey on transfer learning. IEEE Transaction on Knowledge Discovery and Data Engineering, 22 (10). IEEE press.

Panchal, G., Ganatra, A., Kosta, Y., & Panchal, D. (2011). Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, *3*(2), 332–337.

Parasher, M., Sharma, S., Sharma, A., & Gupta, J. (2011). Anatomy on pattern recognition. *Indian Journal of Computer Science and Engineering (IJCSE)*, *2*(3), 371–378.

Pardasani, A., Indi, P. N., Banerjee, S., Kamal, A., & Garg, V. (2019). Smart Assistive Navigation Devices for Visually Impaired People. *2019 IEEE 4th International Conference on*

Computer and Communication Systems (ICCCS), 725–729. doi:10.1109/ccoms.2019.8821654

Paris Workshop, P. in the. (2003). The Paris endoscopic classification of superficial neoplastic lesions: esophagus, stomach, and colon: November 30 to December 1, 2002. *Gastrointest Endosc*, *58*(s6).

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 165–174).

Peng, H., Li, B., Xiong, W., Hu, W., & Ji, R. (2014). Rgbd salient object detection: a benchmark and algorithms. *European conference on computer vision* (pp. 92–109). Springer.

Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., & Geiger, A. (2020). Convolutional occupancy networks. *European Conference on Computer Vision* (pp. 523–540). Springer.

Perazzi, F., Krahenbuhl, P., Pritch, Y., & Sorkine-Hornung, A. (2012). Saliency Filters: Contrast Based Filtering for Salient Region Detection. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 733–740). doi:10.1109/CVPR.2012.6247743

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Piccialli, F., Di Somma, V., Giampaolo, F., Cuomo, S., & Fortino, G. (2021). A survey on deep learning in medicine: Why, how and when? *Information Fusion*, *66*, 111–137.

Poggi, M., & Mattoccia, S. (2016). A wearable mobility aid for the visually impaired based on embedded 3D vision and deep learning. *2016 IEEE Symposium on Computers and Communication (ISCC)* (pp. 208–213). doi:10.1109/ISCC.2016.7543741

Pogorelov, K., Randel, K. R., Griwodz, C., Eskeland, S. L., Lange, T. de, Johansen, D., Spampinato, C., et al. (2017). KVASIR: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection. *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17 (pp. 164–169). Taipei, Taiwan: ACM. doi:10.1145/3083187.3083212

Poirson, A. B., & Wandell, B. A. (1996). Pattern—color separable pathways predict sensitivity to simple colored patterns. *Vision research*, *36*(4), 515–526.

Prior, F., Smith, K., Sharma, A., Kirby, J., Tarbox, L., Clark, K., Bennett, W., et al. (2017). The public cancer radiology imaging collections of The Cancer Imaging Archive. *Scientific data*, *4*, 170124.

Provost, F., & Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions In: Proc of the 3rd International Conference on Knowledge Discovery and Data Mining.

Qin, X., Zhang, Z., Huang, C., Gao, C., Dehghan, M., & Jagersand, M. (2019). BASNet: Boundary-Aware Salient Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qu, L., He, S., Zhang, J., Tian, J., Tang, Y., & Yang, Q. (2017). RGBD salient object detection via deep fusion. *IEEE Transactions on Image Processing*, *26*(5), 2274–2285.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Rahman, M. M., Islam, M. M., Ahmmed, S., & Khan, S. A. (2020). Obstacle and fall detection to guide the visually impaired people with real time monitoring. *SN Computer Science*, *1*, 1–10.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Reed, A. W., Kim, H., Anirudh, R., Mohan, K. A., Champley, K., Kang, J., & Jayasuriya, S. (2021). Dynamic ct reconstruction from limited views with implicit neural representations and parametric motion fields. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 2258–2268).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* (pp. 91–99).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Riche, N., Duvinage, M., Mancas, M., Gosselin, B., & Dutoit, T. (2013). Saliency and human fixations: State-of-the-art and study of comparison metrics. *Proceedings of the IEEE international conference on computer vision* (pp. 1153–1160).

Rivest, J.-F., Soille, P., & Beucher, S. (1993). Morphological gradients. *J. Electronic Imaging*, *2*, 326–336. doi:10.1117/12.159642

Rivest, R. L. (1987). Learning decision lists. *Machine learning*, *2*(3), 229–246.

Rodrıguez, A., Bergasa, L. M., Alcantarilla, P. F., Yebes, J., & Cela, A. (2012). Obstacle avoidance system for assisting visually impaired people. *Proceedings of the IEEE Intelligent Vehicles Symposium Workshops, Madrid, Spain* (Vol. 35, p. 16).

Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Ron, K., & George, H. J. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273–324.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241). Springer.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

Rosenblatt, F. (1958). *Two theorems of statistical separability in the perceptron*. United States Department of Commerce.

Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the IRE*, *48*(3), 301–309.

Ruby, U., & Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *Int. J. Adv. Trends Comput. Sci. Eng*, *9*(10).

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2022). Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, *16*, 1–85.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation*. California Univ San Diego La Jolla Inst for Cognitive Science.

Russell, S. J. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in neural information processing systems*, *30*.

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. *Proceedings of the 2000 symposium on Eye tracking research & applications* (pp. 71–78). ACM.

Sandri, M., & Zuccolotto, P. (2006). Variable selection using random forests. *Data analysis, classification and the forward search* (pp. 263–270). Springer.

Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, *18*(4), 80–92.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *International conference on artificial neural networks* (pp. 92–101). Springer.

Schmidhuber, J. (2007). Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity. *International conference on discovery science* (pp. 26–38). Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Schmidhuber, J., Hochreiter, S., & others. (1997). Long short-term memory. *Neural Comput*, *9*(8), 1735–1780.

Schoen, R. E., Gerber, L. D., & Margulies, C. (1997). The pathologic measurement of polyp size is preferable to the endoscopic estimate. *Gastrointestinal endoscopy*, *46*(6), 492–496.

Schwarze, T., Lauer, M., Schwaab, M., Romanovas, M., Böhm, S., & Jürgensohn, T. (2016). A camera-based mobility aid for visually impaired people. *KI-Künstliche Intelligenz*, *30*(1), 29–36.

Selbst, A., & Powles, J. (2018). Meaningful Information" and the Right to Explanation. *Conference on Fairness, Accountability and Transparency* (pp. 48–48). PMLR.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., & Batra, D. (2016). Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR*, *abs/1610.02391*. Retrieved from http://arxiv.org/abs/1610.02391

Shah, S. A. A. (2019). Spatial hierarchical analysis deep neural network for RGB-D object recognition. *Pacific-Rim Symposium on Image and Video Technology* (pp. 183–193). Springer.

Shahira, K., Tripathy, S., & Lijiya, A. (2019). Obstacle detection, depth estimation and warning system for visually impaired people. *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)* (pp. 863–868). IEEE.

Shapiro, L. G., Stockman, G. C., & others. (2001). *Computer vision* (Vol. 3). Prentice Hall New Jersey.

Sharma, A., & Mishra, P. K. (2022). Covid-MANet: Multi-task attention network for explainable diagnosis and severity assessment of COVID-19 from CXR images. *Pattern Recognition*, 108826.

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning Important Features Through Propagating Activation Differences. *CoRR*, *abs/1704.02685*. Retrieved from http://arxiv.org/abs/1704.02685

Shurrab, S., & Duwairi, R. (2022). Self-supervised learning methods and applications in medical imaging analysis: A survey. *PeerJ Computer Science*, *8*, e1045.

Siegel, R. L., Miller, K. D., Fuchs, H. E., & Jemal, A. (2021). Cancer statistics, 2021. *CA: a cancer journal for clinicians*, *71*(1), 7–33.

Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. *European conference on computer vision* (pp. 746–760). Springer.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sinha, T., Verma, B., & Haidar, A. (2017). Optimization of convolutional neural network parameters for image classification. *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1–7). IEEE.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., & Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, *33*, 7462–7473.

Slama, C. C. (1980). *Manual of Photogrammetry.* America Society of Photogrammetry,.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

Society, R. (2017). *Machine Learning: The Power and Promise of Computers that Learn by Example: an Introduction*. Royal Society.

Song, H., Liu, Z., Du, H., & Sun, G. (2016). Depth-aware saliency detection using discriminative saliency fusion. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1626–1630). IEEE.

Song, Y., Chen, S., Zhao, Y., & Jin, Q. (2019). Unpaired cross-lingual image caption generation with self-supervised rewards. *Proceedings of the 27th ACM International Conference on Multimedia* (pp. 784–792).

Steponnait, V. K. (2016). Reform of EU data protection rules. *Reform of EU data protection rules, Location: Vilnius, Lithuania*.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Cardoso, M. J. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Deep*

learning in medical image analysis and multimodal learning for clinical decision support (pp. 240–248). Springer.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic Attribution for Deep Networks. *CoRR*, *abs/1703.01365*. Retrieved from http://arxiv.org/abs/1703.01365

Suresh, A., Arora, C., Laha, D., Gaba, D., & Bhambri, S. (2017). Intelligent Smart Glass for Visually Impaired Using Deep Learning Machine Vision Techniques and Robot Operating System (ROS). *International Conference on Robot Intelligence Technology and Applications* (pp. 99–112). Springer.

Sutton, R. (1986). Two problems with back propagation and other steepest descent learning procedures for networks. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986* (pp. 823–832).

Suzuki, S., & others. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, *30*(1), 32–46.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Tapu, R., Mocanu, B., & Zaharia, T. (2017). DEEP-SEE: Joint Object Detection, Tracking and Recognition with Application to Visually Impaired Navigational Assistance. *Sensors*, *17*(11), 2473.

Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). Cnn-slam: Real-time dense monocular slam with learned depth prediction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6243–6252).

Tektonidis, M., & Monnin, D. (2020). Monocular depth estimation for vision-based vehicles based on a self-supervised learning method. *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2020* (Vol. 11415, p. 114150C). International Society for Optics and Photonics.

Thunström, A. O., & Steingrimsson, S. (2022). Can GPT-3 write an academic paper on itself, with minimal human input?

Triantafyllou, G., Dimas, G., Kalozoumis, P., & Iakovidis, D. K. (2022). Reconstruction of Cultural Heritage 3D Models from Sparse Point Clouds Using Implicit Neural Representations. *International Conference on Pattern Recognition*. Springer.

Tuceryan, M., & Jain, A. K. (1993). Texture analysis. *Handbook of pattern recognition and computer vision*, 235–276.

Uzair, M., & Jamil, N. (2020). Effects of hidden layers on the efficiency of neural networks. *2020 IEEE 23rd international multitopic conference (INMIC)* (pp. 1–6). IEEE.

Visentini-Scarzanella, M., Kawasaki, H., Furukawa, R., Bonino, M. A., Arolfo, S., Lo Secco, G., Arezzo, A., et al. (2018). A structured light laser probe for gastrointestinal polyp size measurement: a preliminary comparative study. *Endoscopy International Open*, *06*(05), E602–E609. doi:10.1055/a-0577-2798

Wang, C., & Mahadevan, S. (2011). Heterogeneous domain adaptation using manifold alignment. *Twenty-second international joint conference on artificial intelligence*.

Wang, N., & Gong, X. (2019). Adaptive fusion for RGB-D salient object detection. *IEEE Access*, *7*, 55277–55284.

Wang, Z., Vandersteen, C., Demarcy, T., Gnansia, D., Raffaelli, C., Guevara, N., & Delingette, H. (2020). A Deep Learning based Fast Signed Distance Map Generation. *arXiv preprint arXiv:2005.12662*.

Webots. (n.d.). http://www.cyberbotics.com. (C. Ltd., Ed.). Retrieved from http://www.cyberbotics.com

WHO. (2018). World Health Organization - Blindness and visual impairement. Retrieved from http://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., et al. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (pp. 38–45).

Woniak, M., Grana, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*, 3–17.

Wu, Q., Li, Y., Xu, L., Feng, R., Wei, H., Yang, Q., Yu, B., et al. (2021). Irem: High-resolution magnetic resonance image reconstruction via implicit neural representation. *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 65–74). Springer.

Wyszecki, G., & Stiles, W. S. (2000). *Color Science: Concepts and Methods, Quantitative Data and Formulae 2nd Edition*. Wiley-Interscience.

Yadav, S., Joshi, R. C., Dutta, M. K., Kiac, M., & Sikora, P. (2020). Fusion of Object Recognition and Obstacle Detection approach for Assisting Visually Challenged Person. *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)* (pp. 537–540). IEEE.

Yan, J., He, G., Basiri, A., & Hancock, C. (2019). 3-D passive-vision-aided pedestrian dead reckoning for indoor positioning. *IEEE Transactions on Instrumentation and Measurement*, *69*(4), 1370–1386.

Yang, G., Ye, Q., & Xia, J. (2022). Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. *Information Fusion*, *77*, 29–52.

Yang, K., Wang, K., Zhao, X., Cheng, R., Bai, J., Yang, Y., & Liu, D. (2017). IR stereo realsense: decreasing minimum range of navigational assistance for visually impaired individuals. *Journal of Ambient Intelligence and Smart Environments*, *9*(6), 743–755.

Yang, Z., Zhang, A., & Sudjianto, A. (2020). GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. *arXiv preprint arXiv:2003.07132*.

Yang, Z., Zhang, A., & Sudjianto, A. (2021). GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. *Pattern Recognition*, *120*, 108192.

Yao, K., Cao, F., Leung, Y., & Liang, J. (2021). Deep neural network compression through interpretability-based filter pruning. *Pattern Recognition*, *119*, 108056.

Yen, J.-C., Chang, F.-J., & Chang, S. (1995). A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, *4*(3), 370–378.

Yu, L., Xiang, W., Fang, J., Chen, Y.-P. P., & Zhu, R. (2022). A novel explainable neural network for Alzheimer's disease diagnosis. *Pattern Recognition*, *131*, 108876.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, D., Han, J., Han, J., & Shao, L. (2015). Cosaliency detection based on intrasaliency prior transfer and deep intersaliency mining. *IEEE transactions on neural networks and learning systems*, *27*(6), 1163–1176.

Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, *26*(7), 3142–3155.

Zhang, Q., Wu, Y. N., & Zhu, S.-C. (2018). Interpretable convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8827–8836).

Zhang, Q., Yang, Y., Ma, H., & Wu, Y. N. (2019). Interpreting cnns via decision trees. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6261–6270).

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. *Proceedings of the Seventh IEEE International Conference on Computer Vision* (Vol. 1, pp. 666–673 vol.1). doi:10.1109/ICCV.1999.791289

Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, *31*.

Zhao, J.-X., Cao, Y., Fan, D.-P., Cheng, M.-M., Li, X.-Y., & Zhang, L. (2019). Contrast prior and fluid pyramid integration for RGBD salient object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3927–3936).

Zheng, F., Tang, H., & Liu, Y.-H. (2018). Odometry-vision-based ground vehicle motion estimation with se (2)-constrained se (3) poses. *IEEE transactions on cybernetics*, *49*(7), 2652–2663.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2921–2929).

Zhou, J. T., Tsang, I. W., Pan, S. J., & Tan, M. (2014). Heterogeneous domain adaptation for multiple classes. *Artificial intelligence and statistics* (pp. 1095–1103). PMLR.

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. *Deep learning in medical image analysis and multimodal learning for clinical decision support* (pp. 3–11). Springer.

Zhu, R., Yang, X., Hold-Geoffroy, Y., Perazzi, F., Eisenmann, J., Sunkavalli, K., & Chandraker, M. (2020). Single View Metrology in the Wild. *arXiv preprint arXiv:2007.09529*.