Tampere University

BISHWO PRAKASH ADHIKARI

Computer Assisted Image Labeling for Object Detection Using Deep Learning

Tampere University Dissertations 608

Tampere University Dissertations 608

BISHWO PRAKASH ADHIKARI

Computer Assisted Image Labeling for Object Detection Using Deep Learning

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the Auditorium TB109
of the Tietotalo, Korkeakoulunkatu 1, Tampere,
on 10 June 2022, at 12 o'clock

ACADEMIC DISSERTATION
Tampere University,
Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor and Custos
Associate Professor
Esa Rahtu
Tampere University
Finland

Supervisor
Dr. Heikki Huttunen
Visy Oy
Finland

Pre-examiners
Assistant Professor
Li Liu
University of Oulu
Finland

Associate Professor
Said Pertuz
Universidad Industrial de Santander
Columbia

Opponent
Associate Professor
Miguel Bordallo López
University of Oulu
Finland

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Cover design: Roihu Inc.

ISBN 978-952-03-2419-3 (print)
ISBN 978-952-03-2420-9 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-2420-9

PunaMusta Oy – Yliopistopaino
Joensuu 2022

PREFACE

The study of this thesis was conducted at Tampere University during 2018 – 2021. The financial support from the Business Finland project MIDAS and Tampere University ITC faculty support are greatly acknowledged.

First and foremost, I would like to extend my sincere gratitude to my supervisors, Associate Prof. Esa Rahtu and Dr. Heikki Huttunen, for their constant guidance, unparalleled support, excellent supervision, and great suggestions throughout my research career. I am forever thankful to Heikki for accepting me into his research group and introducing this amazing research world. I am very grateful to Esa for his precise instructions and persistent encouragement. It would have been more challenging to complete this dissertation without their unwavering guidance.

Assoc. Prof. Said Pertuz and Assist. Prof. Li Liu are kindly acknowledged for evaluating this dissertation. I would like to extend my gratitude to Assoc. Prof. Miguel Bordallo López for agreeing to act as my opponent.

I want to thank all co-authors, including Helena Leppäkoski, Jukka Yrjänäinen, Jukka Peltomäki, and Xingyang Ni for their valuable contributions and suggestions. I want to thank everyone from the former machine learning group for the amazing working environment and great discussions. Especially, I would like to thank Saeed Bakhshi Germi, Xingyang Ni and Yi Zhou for sharing excellent work environment, great discussions, and friendships.

Last but not least, I want to thank my parents and siblings. Without their constant love, support and encouragement, I would have not been able to come this far and complete this journey.

Tampere, April 2022

Bishwo Prakash Adhikari

ABSTRACT

Deep learning-based object detectors have shown outstanding performance with state-of-the-art results on public benchmarks. However, they typically consist of millions of parameters and require a large number of training samples to tune these parameters appropriately. These samples are labeled by human annotators, which is a tedious, time-consuming, and expensive process. Moreover, object detectors have high computational costs both for the training and inference phase. This dissertation considers these two aspects of training and deploying deep learning object detectors.

First, we study data labeling for the training phase and the robustness of object detectors towards label noise. We classify possible label noise scenarios in 2D object detection and study the sensitivity of one-stage object detectors to label noise in the training phase. We then propose methods for efficient bounding box annotation by utilizing human-machine collaboration. Extensive experiments have been done to study an efficient and effective bounding box annotation scheme for deep learning object detectors. Additionally, we created an easy-to-use, medium-sized, multiclass, fully labeled object detection dataset from indoor premises and released it publicly for registration-free use.

Second, we study the practical problem of object detection network deployment with an efficient implementation of the object detection network for applications such as facial analysis, human detection and tracking, and the path prediction of mobile objects on resource-limited devices. We implemented object detection in an image processing pipeline integrating with other tasks for multiple applications and studied the optimal design process. We present the details of the system-level design to incorporate a multitasking network efficiently with the proper system architecture design.

CONTENTS

ABBREVIATIONS

AI	Artificial Intelligence
AMT	Amazon Mechanical Turk
AP	Average Precision
CNN	Convolutional Neural Networks
DL	Deep Learning
DNN	Deep Neural Networks
et al.	and others, from Latin *et alii*
FDDB	Face Detection Dataset and Benchmark
FPN	Feature Pyramid Network
HOG	Histogram of Oriented Gradients
mAP	mean Average Precision
ML	Machine Learning
NCS	Neural Computing Stick
RCNN	Region-based Convolutional Neural Networks
ResNet	Residual Networks
RoI	Region of Interest
RPN	Region Proposal Network
SSD	Single Shot Multi-box Detector
SVM	Support Vector Machine
WSOL	Weakly Supervised Object Localization

ORIGINAL PUBLICATIONS

Publication I B. Adhikari, J. Peltomaki, J. Puura and H. Huttunen. Faster Bounding Box Annotation for Object Detection in Indoor Scenes. *European Workshop on Visual Information Processing (EUVIP)*. 2018. DOI: 10.1109/EUVIP.2018.8611732.

Publication II B. Adhikari and H. Huttunen. Iterative Bounding Box Annotation for Object Detection. *International Conference on Pattern Recognition (ICPR)*. 2021, 4040–4046. DOI: 10.1109/ICPR48806.2021.9412956.

Publication III B. Adhikari, E. Rahtu and H. Huttunen. Sample Selection for Efficient Image Annotation. *European Workshop on Visual Information Processing (EUVIP)*. 2021. DOI: 10.1109/EUVIP50544.2021.9484022.

Publication IV B. Adhikari, J. Peltomäki, S. B. Germi, E. Rahtu and H. Huttunen. Effect of Label Noise on Robustness of Deep Neural Network Object Detectors. *Computer Safety, Reliability, and Security. SAFECOMP Workshops*. Ed. by I. Habli, M. Sujan, S. Gerasimou, E. Schoitsch and F. Bitsch. 2021, 239–250. DOI: 10.1007/978-3-030-83906-2_19.

Publication V J. Yrjänäinen, X. Ni, B. Adhikari and H. Huttunen. Privacy-Aware Edge Computing System For People Tracking. *IEEE International Conference on Image Processing (ICIP)*. 2020, 2096–2100. DOI: 10.1109/ICIP40778.2020.9191260.

Publication VI B. Adhikari, X. Ni, E. Rahtu and H. Huttunen. Towards a Real-time System for Facial Analysis. *IEEE International Workshop on*

Multimedia Signal Processing (MMSP). 2021. DOI: 10.1109/MMSP53017.2021.9733663.

Publication VII H. Leppäkoski, B. Adhikari, L. Raivio and R. Ritala. Prediction of Future Paths of Mobile Objects Using Path Library. *Open Engineering* 11.1 (2021), 1048–1058. DOI: 10.1515/eng-2021-0103.

Author's contribution

The current author's contribution to each publication is as follows.

Publication I — The author was responsible for the data collection, annotation, experiments and wrote the manuscript. The co-authors provided valuable insights for the experiments and manuscript.

Publication II — The author was responsible for implementation, experiments and wrote the manuscript as the corresponding author.

Publication III — The author was responsible for implementation, experiments and wrote the manuscript as the corresponding author.

Publication IV — The author was responsible for conducting experiments, writing and revising the manuscript. The co-authors participated in writing the manuscript and provided valuable comments and suggestions.

Publication V — The author participated in object detection experiments and wrote the relevant section in the manuscript. In particular, the author trained and fine-tuned multiple variants of object detectors on multiple datasets. The co-authors were responsible for other parts of the experiments and writing the corresponding sections.

Publication VI — The author conducted detection, alignment, and multitask experiments, wrote and revised the manuscript as the corresponding author. The co-authors conducted some experiments and wrote respective sections. The supervisor provided architecture design and valuable insights for the experiments.

Publication VII — The author collected and labeled the data, trained object detection networks, and took part in writing the manuscript. The author designed the system, deployed the detection network for live detection, and gathered data for the experiments. The co-authors were responsible for other parts of the experiment and writing the corresponding sections.

1 INTRODUCTION

With the computers being present in every aspect of daily human life, the endeavor to maximize their potential usage leads us towards Artificial Intelligence (AI) and Machine Learning (ML). Machine Learning has the been a significant part of transforming industry, logistics, medicine, agriculture, and automobiles.

By combining Computer vision and Machine learning, we create a goal of developing systems that can understand the visual content in an image or a video in the same way humans do. The recent evolution in Convolutional Neural Networks (CNN) and deep learning have changed the landscape of this field with significant increase in the performance accuracy compared to traditional methods.

In a typical case, deep object detectors process a large number of examples to understand the underlying features from the training data and apply the logic on unseen samples to predict the output. Collecting such data in large number of images and videos is tedious and expensive. On the other hand, deep networks usually perform better the bigger they get, which means they have large memory footprints. This thesis studies both aspects; efficient object labeling on training dataset for deep learning object detectors and efficient implementation of such networks on image processing pipeline for the real-time performance on low-resourced devices. We aim to propose annotator-friendly human-in-the-loop schemes to facilitate image annotation and experiment with efficient deployment techniques for detection networks on low-resourced devices.

1.1 Objectives and Scope of the Thesis

Supervised object detection is the most common approach widely used in research projects and applications. It is considered a safe choice as a building blocks for higher-level decision-making systems for private and commercial uses. However, for supervised machine learning, the challenge lies in finding a suitable training dataset. Ob-

ject detection datasets consist of images with one or more objects in a frame that are typically assigned by human annotators. As stated before, collecting such a dataset is not straightforward since it requires special attention and domain expertise (e.g., medical images) which costs time and money. As an example, the commonly used crowd-sourcing platform of Amazon Mechanical Turk (AMT) requires 42 seconds per instance to draw a bounding box with a class label on the popular object detection dataset for ImageNet Large Scale Visual Recognition Challenge (LSVRC) [77]. Due to these issues scientists started exploring alternative annotation schemes.

Researchers have been resolving this challenge by open-sourcing large-scale datasets, developing efficient annotation tools, and studying efficient alternatives such as transfer learning and weakly supervised learning. Many fully labeled public benchmark datasets from multiple domains have facilitated object detection breakthroughs in various fields. However, for real-world detection applications, these public datasets rarely satisfy the need for good performing object detector. Object detection tasks can be particular to a specific environment and use case, so relying only on public datasets often does not generalize well. Thus, there is a need for custom datasets from the testing environment. The custom dataset creation process can be enhanced with the help of in-house, efficient, and human-friendly data labeling tools, leading to broader usage of object detection-based applications.

Moreover, modern CNN-based object detectors have millions of parameters that require powerful computing resources to train and inference for the applications. The broader availability of mobile and embedded devices have initiated research towards more efficient implementation of detectors for real-time inference on these devices.

The objective of this thesis is to improve the efficiency of manual labeling with the assistance of a machine learning tool for bounding box annotation. In addition to annotator-friendly annotation schemes, the thesis aims to implement object detectors for vision-based applications and find the optimal design strategy for deploying on imaging pipelines for low-powered edge devices at scale.

1.2 Research Questions

In this dissertation, there are mainly three perspectives from which the research questions are derived and studied. The first research perspective is about the Label noise

impact on CNN-based object detectors. The quality of training data is one of the critical factors affecting the performance of machine learning models. Since label noise in the training dataset is inevitable, it is worth investing in the techniques to tolerate the noise while achieving the desired performance.

The second research perspective is about efficient image labeling for modern object detection. The annotation of object location with a class category on image datasets is costly and time-consuming. While there are many services and platforms available for data annotation, the cost, privacy issues, and the nature of the data sometimes make it impossible to use such services. For example, companies usually do not want to share their data to protect confidentiality and competence. Thus, an annotator-friendly easy to adopt method is needed for object detection.

The third research perspective is about the efficient implementation of CNN-based object detectors in diverse applications. CNN-based object detectors have been successful with state-of-the-art performance in different domains. Typically, these detectors consist of millions of parameters and require extensive resources to store and operate. On the other hand, the rapid growth of mobile phones, embedded devices, and cloud computing has expanded the use of detection-based systems in applications designed to operate on these platforms. The speed versus accuracy or memory footprint versus detection performance trade-off is a long-standing debate in deep learning. The existing detection network can be efficiently deployed by applying tricks such as the network pruning and modifying the network parameters such as input size, number of layers, and feature extraction network.

Based on the perspectives mentioned above, we attempt to answer the following research questions studied in this thesis.

- *Research Question 1:*
  How sensitive existing object detectors are to label noise and how to improve the robustness of the detectors?
- *Research Question 2:*
  How to use existing machine learning and human annotator efficiently to label training data for supervised object detectors?
- *Research Question 3:*
  How to integrate object detection networks into an image-processing pipeline for the resource-limited platforms and put them into production at scale?

1.3 Summary of the Publications

The results and the contributions of the publications included in this thesis are summarized as follows.

Publication I — This article proposes a semi-automatic approach to speed up the bounding box annotation process with a minimal manual workload. Our approach trains the object detector on a small set of manually labeled data and learns to predict bounding boxes and class labels for the unlabeled data. A human annotator then reviews these annotation proposals and fixes potential mistakes. We showed that the proposed approach could reduce manual annotation workload by up to 90%. Additionally, we published a fully labeled indoor dataset for the object detector training.

Publication II — We propose an iterative annotation method that utilizes human-machine collaboration with human-in-the-loop annotation. At each iteration, the method trains an object detection network on a recently labeled dataset and predicts annotation proposals for the next batch of unlabeled images. These predictions are reviewed and verified by a human annotator. The iterative annotation is straightforward and it could reduce manual annotation workload by up to 80%, depending on the sample selection strategies.

Publication III — This work proposed an efficient sample selection approach that provides informative samples to label with our iterative annotation method proposed in publication II. We aimed to label a whole dataset by selecting informative images based on a practical and intuitive idea. We showed that the image-to-image similarity metric is effective for selecting image samples for efficient annotation of object detection datasets.

Publication IV — This paper studies the robustness of two different object detection loss functions on label noise. Experiments with different amounts of missing labels showed that focal loss is more sensitive

than cross-entropy loss. Also, the experiments showed that suitable hyperparameters for the focal loss function could improve the detector robustness even in extreme label noise cases.

Publication V — This article studies challenges in the practical implementation of a distributed computer vision system for person detection and tracking. A privacy-aware system is designed and tested in a real-world environment while receiving data from multiple edge devices via the cloud server. Additionally, the design and implementation aspects for real-time execution in a low-resourced edge computing device are studied.

Publication VI — This article investigates practical implementation issues of multiple algorithms for facial attributes recognition tasks integrated within a single system. The system utilizes known techniques that recognize age, gender, expression, and face similarity against a celebrity dataset. Moreover, the paper presents a real-time facial analysis system running on a single desktop and evaluates many design options, such as what neural network models to use for individual facial analysis tasks.

Publication VII — This article introduces an efficient object detection system to predict the trajectory of moving objects. A lightweight object detector is trained on custom data collected from the test environment and deployed in a Raspberry Pi equipped with a camera module to detect the moving objects. The proposed approach predicts the trajectory of moving objects based on the detection results and timestamps of the detection.

Structure of the Thesis

The structure of the dissertation is as follows. Chapter 2 presents a brief overview of the field, including machine learning and deep learning, as well as a brief review of the evolution of object detectors and standard object detection networks. Additionally, Chapter 2 provides an overview of different types of image labeling practices used for object recognition tasks, performance measurement metrics, and the dataset used for experiments. Chapter 3 discusses the label noise effect on the datasets while training deep learning object detectors. It summarizes the works on faster and efficient image annotation methods for 2D object detection, namely bounding box annotation. Chapter 4 presents the author works on implementing object detection networks on various imaging pipelines. Finally, Chapter 5 concludes the thesis with discussions about current and possible future research in the field.

2 BACKGROUND

This chapter includes a brief discussion of machine learning, deep neural networks, and object detection to understand the work done in this thesis. The chapter covers some common object detection networks structures, image labeling for deep learning detectors, and evaluation metrics and datasets used in this thesis.

2.1 Machine Learning

Machine learning (ML) is a field of computer science, a subfield of Artificial Intelligence (AI), where the algorithm aims to learn a pattern from the training data to predict information about unseen test data. In some cases, humans can manually design the system to look for specific features to find the pattern. However, this task gets complex exponentially when dealing with more challenging problems. Thus, ML algorithms are evolved to find their own features without any human interference.

With the integration of digital devices such as smartphones and wearable devices in our daily lives, ML is being used in various places without realizing it. Some of the common uses of ML algorithms can be seen in search engines, spam filtering in emails, language translation services, voice assistance, recommendation system, chatbot, etc. Moreover, ML algorithms can extract valuable insight and analyze the data produced and collected from our everyday activities on digital devices and the internet.

Based on the learning principles, ML algorithms are broadly classified into four categories. A brief discussion on each type of learning is presented next.

Supervised Learning

Supervised learning algorithms are trained with labeled data samples and used to create an accurate prediction model under supervision. This model learns from a given training set (input data) and predicts the correct label for testing set. Typical dataset for training supervised learning algorithm consists of a set of input-output pairs. For input variables ($X = \{x_1, x_2, .., x_n\}$) and output variables ($Y = \{y_1, y_2, .., y_n\}$), supervised learning algorithm uses a mapping function $\boldsymbol{y} = f(\boldsymbol{x})$ to map the relation between input-output pair. Supervised learning aims to approximate the mapping function $f(\boldsymbol{x})$ for any new input data $\boldsymbol{x}'$, and the model predicts output label $\boldsymbol{y}'$ for it.

Supervised learning is by far the most applied machine learning method in diverse tasks and applications. Classification and regression are the most common examples of supervised learning. In classification, the goal is to assign input points to one of the class categories of input data, e.g., identifying cats and dogs from images, identifying handwritten digits, and email spam filters. In regression, the goal is to estimate a relationship between a dependent variable and one or more independent variables by predicting output in a numerical quantity, often a continuous variable. House price prediction based on house features, sales prediction based on sales data, and stock price prediction are common regression examples. Classification and regression differ only by their output; the former predicts categorical labels while the latter predicts the numeric value. Some examples of supervised learning are linear regression, logistic regression, decision trees, naive Bayes classifier, nearest neighbor clustering, and support vector machines (SVM).

Unsupervised Learning

Unsupervised learning is learning without predefined labels or direct supervision. Unsupervised learning aims to determine the hidden structure or pattern in a collection of unlabeled data. Clustering is a typical example of unsupervised learning where the data are partitioned into groups (clusters) based on similar-looking features. Other examples are visualization, dimension reduction, and association rule learning. Visualization helps to understand the data and underlying features by presenting complex data into appealing 2D or 3D graphical formats. Dimension reduc-

tion helps to reduce the correlated features from the data by merging into smaller dimensions without losing valuable information. Association rule learning helps to discover the interesting relation between variables in a large dataset.

Semi-supervised Learning

Semi-supervised learning is a combination of both supervised and unsupervised learning methods. In semi-supervised learning, an algorithm is trained on the partially labeled or incompletely labeled dataset. Collecting a large-scale dataset and labeling it for machine learning algorithms is time and resource-consuming. Meanwhile, acquiring a partially labeled dataset is relatively easy. Semi-supervised learning can be used as an alternative option in the case of incomplete annotation or noisy data. A typical example of semi-supervised learning is a text document classifier, where the algorithm learns from a small amount of labeled text data and can still classify a large amount of unlabeled text documents.

Reinforcement Learning

Reinforcement learning is learning of what action to perform and how to map this action to the situations to maximize the reward signal and complete the given task [84]. The learner is asked to perform a task without providing explicit instructions; instead, it must discover its actions to yield maximum reward by trial and error. The learner agent learns from its past experiences, must have a sense of the current environment state, and can take action accordingly to accomplish its goal. Although supervised and unsupervised learning covered most types of learning, they are not adequate for learning from interaction. In interactive problems, it is nontrivial to collect correct and well-representing examples of desired behaviors. In these situations, the learner must learn to form its own experiences and keep improving its performance. Reinforcement learning aims to maximize the reward function rather than trying to find the underlying patterns. Reinforcement learning has been implemented in many real-world applications, especially in autonomous driving, machine vision, and robotics.

2.2 Deep Learning

Deep learning (DL) is a machine learning technique constructed based on artificial neural networks inspired by the structure and function of the brain. DL framework consists of multiple layers of simple modules used for learning and computing nonlinear input-output mapping. The multi-layer model architecture extracts the higher-level representation of data and transforms it to multiple levels of abstractions. Typically, DL network consists of many intermediate layers called hidden layers between the input and output layers.

Recently, learning algorithms are getting more accurate than traditional approaches. They are capable of solving many challenging tasks that were considered impossible to do with ML until ten years ago. DL algorithms have been applied extensively in fields such as computer vision, speech recognition, natural language processing, and medical diagnosis. DL started gaining attraction in computer vision after Krizhevsky *et al.* [42] proposed a CNN-based image classifier with high performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [77] in 2012.

In recent years, CNN-based algorithms have been widely used in the research community leading to rapid growth in publications and a wide range of applications [7, 29, 109, 118]. A recent survey showed that deep learning skills dominate the other skillsets for opening jobs in the AI field. The demand for these skills is increasing further as more diverse research institutions and industries are getting involved with it [109].

Traditional ML algorithms are suitable for structured and low dimensional data, while DL algorithms are well served for large-scale, unstructured, high dimensional data and perceptual problems. The non-linear mathematics of these problems led to development of deeper models in DL. This complexity enables the efficient solving of high-dimensional problems such as object recognition, natural language processing, speech/text recognition, and more.

In the past, the challenges for using many-layered networks were computing resources and enormous datasets required to train the model. Publicly available large-scale benchmark datasets, powerful computing devices, general-purpose Graphics Processing Units (GPU's) and Tensor Processing Units (TPU's), and broader research interests played significant roles in wider use of DL methods.

Common CNN Architectures

CNN, a powerful family of artificial neural networks, utilizes convolution operation via several layers to efficiently extract the underlying features of input data. Modern CNN architectures tend to use a large number of convolutional layers (deep networks) followed by one or more fully connected layers. Here we introduce some of the most common CNN architectures selected for experiments in this thesis.

MobileNet

MobileNet is a lightweight deep neural network designed for efficient computing for mobile vision applications *et al.* [32]. The idea behind the MobileNet architecture is the division of standard convolutional filters into two filters. Standard convolutional filter is divided into depthwise and pointwise convolutional filters where the total computational complexity of standard convolution is higher than the combination of the separable convolutions as shown in Figure 2.1. This division is optimized for computation speed. The reduction in accuracy is relatively small in comparison to the standard one.

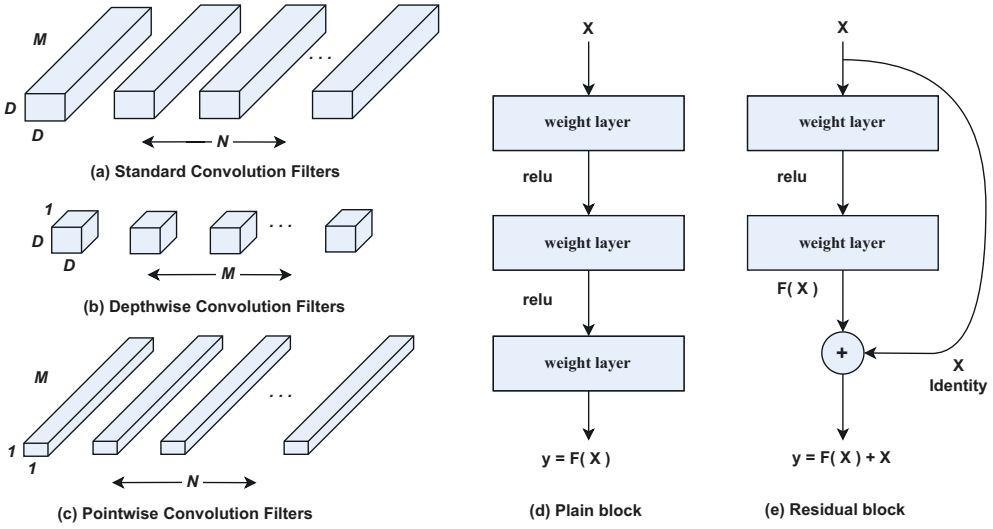


Figure 2.1 Illustration of (a) standard convolution filters, (b) depthwise convolution filters, (c) pointwise convolution filters, (d) regular block in CNN, and (e) residual block of ResNet.

Residual Networks

Residual networks (ResNet) [31] utilizes the idea of learning residuals instead of features in the model. Accuracy degradation happens when the accuracy of a trained model starts dropping after reaching a certain threshold due to overfitting the training data. ResNet was proposed to solve this issue by using the residual function in place of traditional mapping between input and output. In simple terms, a residual function removes learned features from a few layers and connects directly to the output, also known as skip connections. This skip connection benefits the learning process by skipping any particular layer that hurts the network performance. The residual function can be defined as $F(x) = H(x) - x$, where $F(x)$ represents the output of multiple non-linear layers, $H(x)$ represents the mapping function, and x represents the identity function. The largest ResNet consists of 152 residual blocks, while the smallest can be as small as 9 blocks.

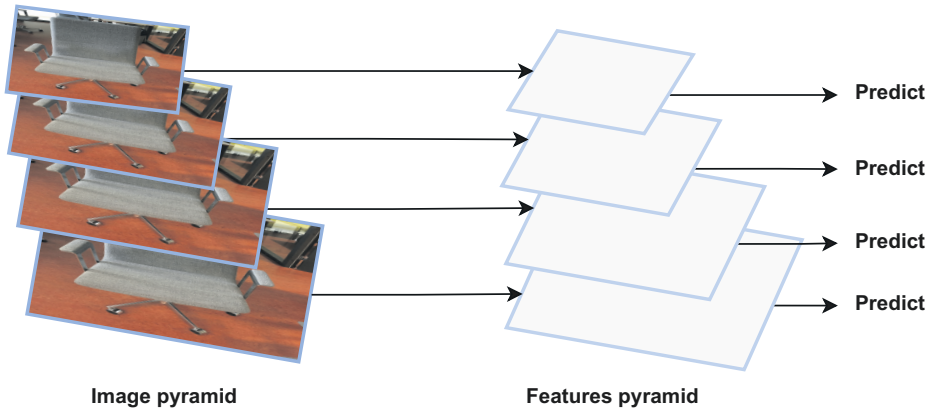


Figure 2.2 Illustration of a featurized image pyramid that uses image pyramid and features pyramid.

Feature Pyramid Network

Feature pyramids are a fundamental component in recognition systems for detecting objects at different scales. However, they are avoided as they are computation and memory intensive. Feature pyramid network (FPN) construct feature pyramids with parallel connections to build high-level semantic feature maps at all scales as illustrated in Figure 2.2.

2.3 Object Detection

In computer vision, image classification is the task of labeling the image as a whole. In addition to labeling, object localization finds the position, rectangle coordinates, of the object. Object detection goes a step ahead of object localization to find all object locations along with the labels. In object detection, the task is to predict a tight bounding box around each object instance and its class label in a test image. Object detection is one of the fundamental and the most challenging task in computer vision [17, 21, 22, 23, 27, 50, 54, 72, 75]. The recognition of object instances is associated with tasks such as classification, localization, object detection, and instance segmentation, as shown in Figure 2.3.

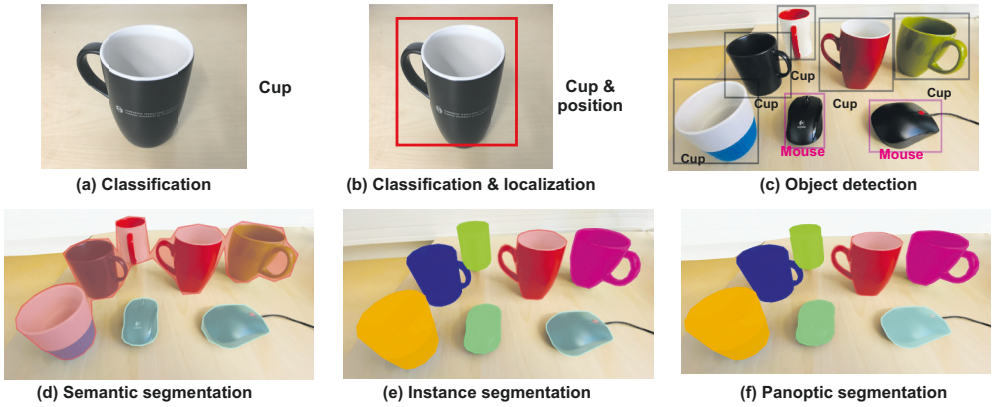


Figure 2.3 The collection of object recognition tasks in computer vision (a) classification, class label of the object in an image, (b) object localization, location of one or more object(s) in an image frame, (c) object detection, class and location of each object in an image, (d) semantic segmentation, every pixel is assigned a label, (e) instance segmentation, precise pixels of each object is labeled, and (f) panoptic segmentation, a combination of semantic segmentation and instance segmentation.

The majority of the modern object detection models are based on supervised learning, where predicting a class category belongs to the classification task and locating object and getting the coordinates belong to the regression task. Additionally, there have been other learning approaches used for object detection, such as semi-supervised and weakly-supervised. The focus of this thesis is mainly on data labeling bottlenecks for implementing supervised object detection.

Commercial applications of object detection such as facial analysis [3, 7], au-

tonomous driving [13, 99, 103], robotics [39], surveillance system [46, 106], and agriculture [41] have seen an increasing trend of object detector usages. Visual recognition applications are rapidly increasing with wide range of accessible datasets, more computing power on devices, advanced imaging hardware and software, and affordable and scaleable cloud services [109].

There is a wide selection of open-sourced libraries and packages for object detection training and deployment. Dlib [38], Detectron [28], OpenCV [62], Pytorch [67], and Tensorflow [89] are commonly used platforms/libraries for modern object detection training and deployment. Recently, researchers have been following the trend of openly publishing their project codes and trained network weight for reference.

History of Object Detectors

The common pattern in object detectors is as follows: a window classifier is trained to detect a presence of a window containing an instance of the target object at training time. The classifier is applied to score these windows on a test image at testing time. The detection is computed by finding the local maximum score.

The very early methods for object detection include appearance-based methods that extract the appearance of local object patches [59, 76, 92]. These methods typically work by extracting a set of interest points from the test image, then finding local features around these points and comparing them with features extracted from the training images using some distance function. Affine-invariant interest point detectors [59, 92] have been extensively used in these schemes and are successful in matching local object regions across images with diverse viewpoints.

After appearance-based methods, the sliding window paradigm was widely used and had shown successful results for object detectors [17, 23, 94]. The core concept of the sliding window methods lies in the position and scale variation of the images. These methods search exhaustively in a dense grid of images and can find the object at any position and scale in the image. A window classifier is trained to detect the window that contains an instance of the target object at training time. At the test time, every window on a dense grid is scored. The detections are the local maximum of the score function.

Dalal and Triggs [17] introduced a very successful feature descriptor called His-

togram of Gradients (HOG) for pedestrian detection. Their HOG person detector uses a sliding window moving around the image by computing the HOG feature descriptor at each position. These features are then used to train a Support Vector Machine (SVM), which classifies inputs as *person* or *not a person*. They also introduced the novel concept of hard negative mining for the training. At first, a model is trained to classify negative windows on training images, the incorrectly classified negative windows are collected and used to train the new model. This process is repetitive till the model converges.

Deformable Part Model (DPM) [23] detector is another widely used object detector that uses multi-scale deformable part models. The DPM is designed on a pictorial structures scheme, *i.e.,* an object can be represented as a collection of parts arranged in a deformable configuration.

Evaluating a large number of sliding windows in an image is computationally heavy. Proposal-based object class detectors are introduced to reduce the number of classifier windows needed to run in an image [8]. Unlike exhaustive search on sliding window approaches, the proposal-based object detector first generates a small number of windows for each image containing all objects of the image. The class-specific window classifier is then evaluated only on this limited pool of proposed windows. Objectiveness measure, a metric that quantifies how likely an object of any class can be in a window, is commonly used as a location prior to the sliding window detector reducing the number of windows needed to evaluate. Researchers proposed many approaches to generate class-specific object proposals detectors [69, 93, 117]

Modern object detectors consist of a backbone that is typically state-of-the-art image classification CNN *e.g.*, VGG [81], ResNet [31], MobileNet [32, 79], EfficientNet [87], etc., and additional layers to predict location and label from backbone features. CNN based methods can be broadly divided into two sub-categories: two-stage [26, 27, 30, 75] and one-stage [50, 54, 73, 74, 95]. These methods have been proven effective with state-of-the-art performance on challenging public benchmark datasets [20, 22, 51, 55].

In contrast to traditional detectors mentioned above, CNN-based object detectors are trained end-to-end to extract the visual context of the image window by learning feature representation with the window classifier instead of hand-crafted features. Recent survey papers [53, 108, 118] nicely summarized the recent develop-

ments in deep learning based object detectors.

Two Stage Detectors

In two-stage object detectors, the detection is performed in two major passes: at first, a set of the region of interests (RoI), a region with a high probability of having an object, is obtained by a selective search algorithm or a region proposal network as illustrated in Figure 2.4. Then classification and regression sub-networks are used for classifying and localizing objects presents in these region proposals.

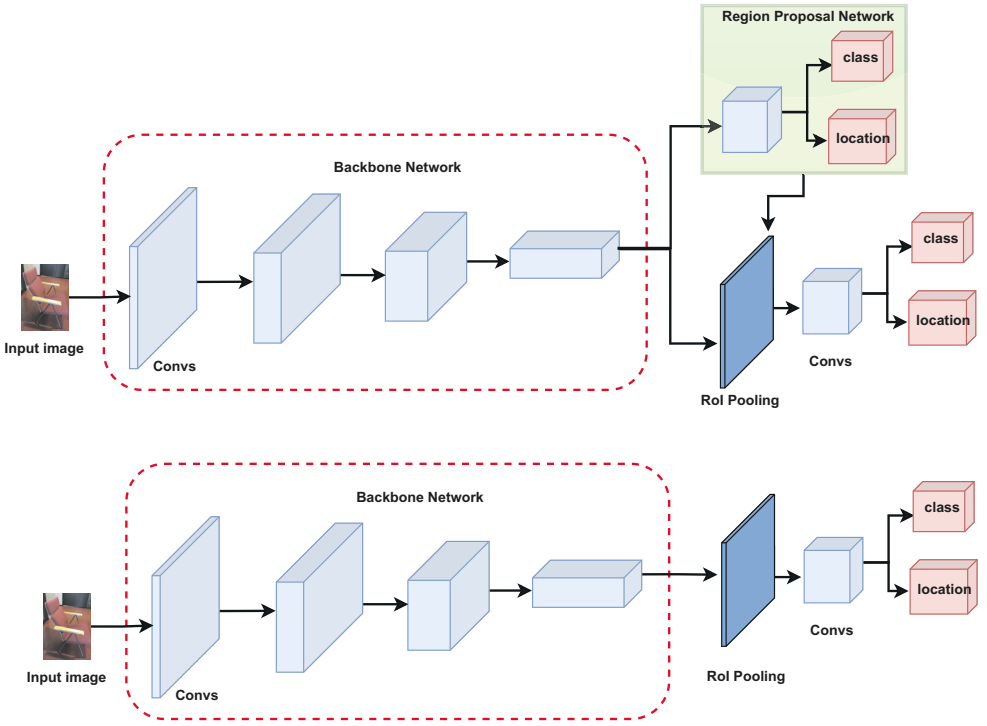


Figure 2.4 Overview of two-stage detector (top) and one-stage detector (bottom) architectures.

Region-based Convolutional Neural Networks (RCNN) families are popular two-stage object detectors and have been widely used in object detection with state-of-the-art performance on public benchmark datasets [26, 30, 75]. RCNN detectors give better detection accuracy but are computationally heavier due to the high number of candidate region proposals for each image at the first stage.

RCNN

In 2014, Grishick *et al.* proposed a region-based convolutional neural network (RCNN) for the object detection task [27]. The RCNN detector workflow is as follows. At first, a selective search is applied to an input image to identifies class independent regions of interest (2k per image) that may contain target objects. The extracted region candidates are wrapped in a fixed size. The pretrained CNN classification network is used to finetune these warped proposals and to extract features. Next, the feature vectors are passed through binary SVMs which are trained for each class independently. Finally, a regression network is trained to correct the predictions of the bounding box location obtained at the first stage. Based on this RCNN principle, many new variants have been proposed over time, surpassing the detection performance with faster inference time [26, 30, 75].

Fast RCNN

Fast RCNN is the improved version of the RCNN detector that unified three independent models into a single joint framework for enhancing training by sharing computations. The Fast RCNN detector workflow is as follows. At first, an input image and a set of object proposals are given. The input image is processed with several convolutions and max-pooling layers to extract the feature map. The RoI pooling layer is applied to extract the feature vector from the feature map. Each extracted feature vector is fed into a fully connected layer connected to two output layers. One produces softmax probability (softmax classifier) and the other predicts four numbers for bounding box positions (box regressor).

Faster RCNN

Faster RCNN is an improved version of Fast RCNN that speeds up the computation in the previous version by replacing the slow selective search algorithm for the region proposal with a fast CNN network. An end-to-end region proposal network (RPN) is used to simultaneously predict object bounds and objectness scores at each position. The Fast RCNN network then utilizes these high-quality region proposals for detection. This change significantly reduced the computation and increased the detection accuracy [75]. The Faster RCNN detector workflow is as follows. At

first, the input image is passed through a CNN network to produce a feature map. These extracted feature maps are sent to RPN and Fast RCNN networks. The RPN predicts region proposals based on the extracted features. The Fast RCNN network is trained on the region proposals generated by the recent RPN. Finally, the unique layers of Fast RCNN are fine-tuned to predict the object's class label and bounding box location.

Mask RCNN

Mask RCNN [30] is the extension of the Faster RCNN detector with a pixel-level image segmentation layer. Mask RCCN is designed on top of the Faster RCNN with an added branch that predicts an object mask in parallel with the existing classification and localization branches. The additional mask branch is a fully connected network applied to all regions of interest to predict segmentation masks for each object. The loss function to train Mask RCNN is the combination of individual task losses: loss of classification, localization, and segmentation mask as
$(L = L_{classification} + L_{box} + L_{mask})$.

The architecture designs of RCNN object detectors mentioned above are illustrated in Figure 2.5.

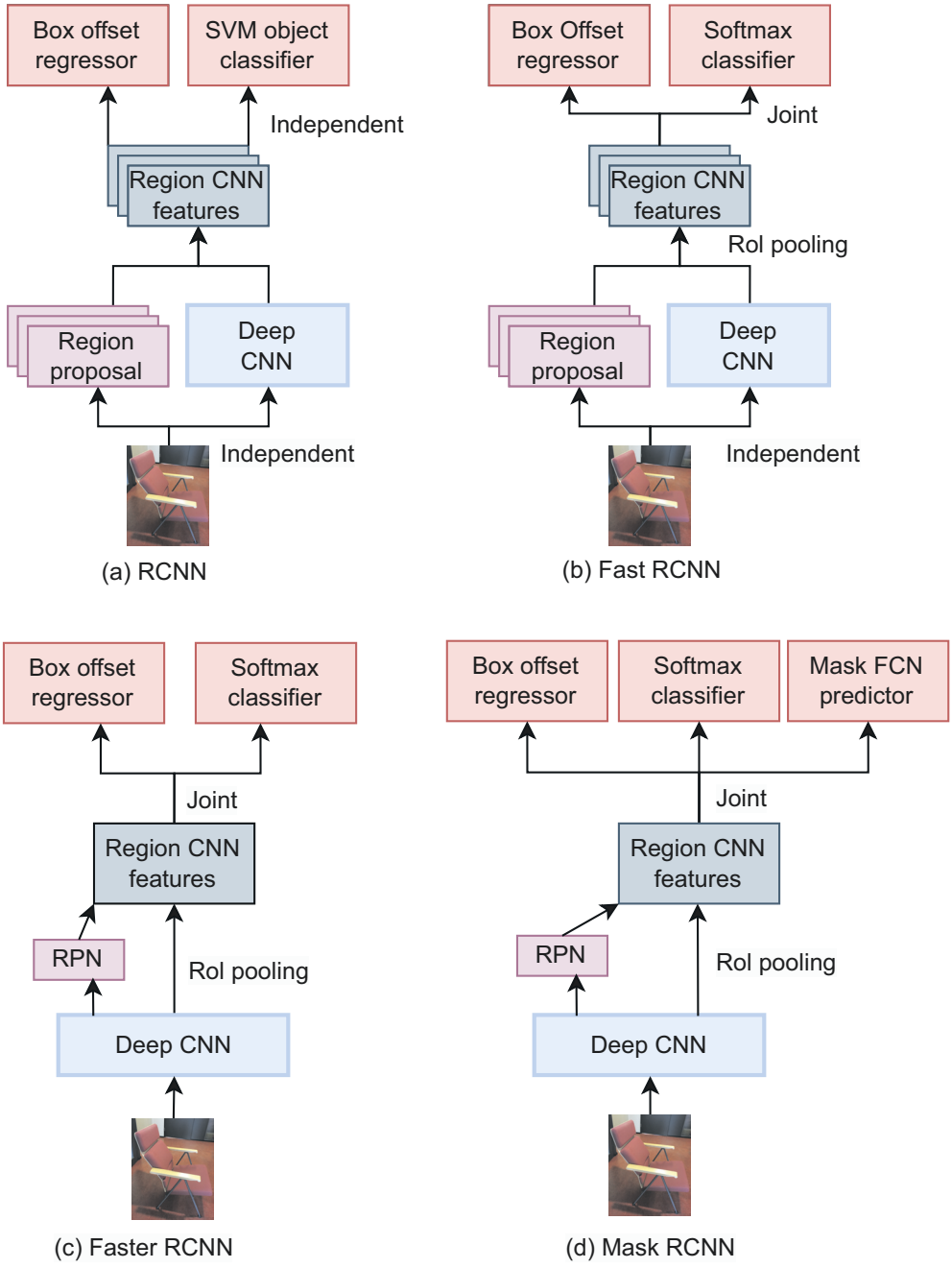


Figure 2.5 Overview of RCNN family detectors: (a) RCNN, (b) Fast-RCNN, (c) Faster-RCNN, and (d) Mask RCNN.

One Stage Detectors

One stage detector generates detection results as a single forward pass, eliminating the need for a separate region proposal stage. A series of convolutional layers produce a collection of bounding box coordinates and scores to make final detections. One-stage detectors are relatively faster and slightly less accurate than two-stage detectors. These detectors are suitable for efficient implementation and have been widely used in applications. In our publications V, VI, and VII, we used one stage object detector for different visual applications.

Single Shot Multibox Detector (SSD)

Single shot multibox detector (SSD) performs the object detection task with a single pass feed-forward CNN [54]. This CNN will propose multiple bounding boxes with specific sizes for objects in the image and class scores for each instance within those boxes. A non-maximum suppression step is applied to eliminate boxes below a certain detection threshold to produce the final detections. Unlike two-stage models, SSD does not generate region proposals or re-samples image segments, saving computational time.

The input image is passed through a series of convolutional layers followed by down-sampling layers as shown in Figure 2.6. The SSD detector workflow is as follows. First, multiple convolutional layers are applied on the input image, which results in a multi-scale set of feature maps. Then a convolutional filter is applied to each feature map to generate a set of bounding boxes (anchor boxes). After that, the bounding box and the probability scores for each class is predicted simultaneously. Finally, a non-maximum suppression is applied to remove non-relevant predicted bounding boxes, resulting in a better match for the bounding boxes.

RetinaNet

RetinaNet is a family of one-stage object detectors that utilize Focal loss as their loss function to deal with the foreground-background class imbalance problem posed in one-stage detectors [50].

While in two-stage detectors, the RPN provides potential regions for the classifier at the second stage to deal with the foreground-background imbalance problem, the

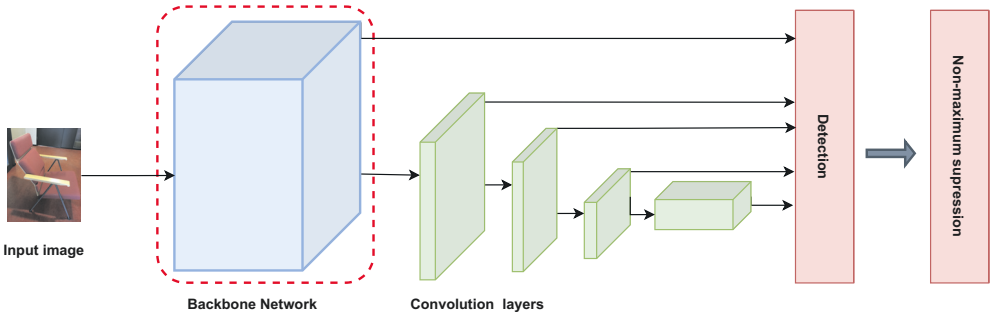


Figure 2.6 SSD uses VGG-16 [81] model as the base (backbone) for extracting image features followed by a series of several convolutional layers with descending size.

one-stage detectors do not have such potential inherently. However, utilizing focal loss instead of cross-entropy in one-stage detectors will result in a higher accuracy than the existing state-of-the-art two-stage detectors while maintaining the speed advantage of one-stage detectors. RetinaNet is based on a *ResNet-101-FPN* backbone with two task-specific headers for classification and localization. As shown in Figure 2.7, this architecture utilizes previously mentioned anchors and feature pyramids.

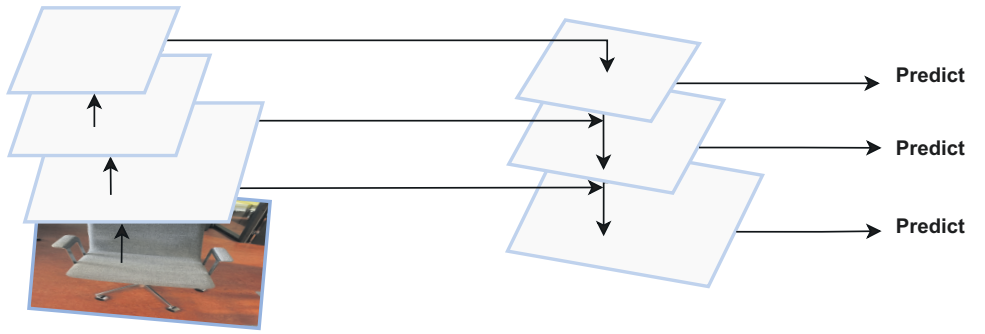


Figure 2.7 Feature pyramid network combines low-resolution features with high-resolution features via top-down pathways and side-wise connections [50].

In addition to above mentioned detection networks there are other detectors heavily used in both research community and in commercial applications [74, 99]. Other most recent state-of-the-art object detection frameworks includes EfficientDet [88], FCOS [90], and YOLOR [95]. Among newer neural network architec-

tures, VGG [81], Inception [85], MobileNet V2 [79], and EfficientNet [87] are some of the other popular networks used as backbone in object detection frameworks.

Challenges in Object Detection

Objects detection tasks can be very challenging depending on the factors affecting the visual appearances of the objects. Objects in the image may contain various viewpoints and pose, be truncated, partially occluded, noisy, blurry, and/or dark. The challenges in object detection are the robustness of the object detectors against such problems, the computational complexity of the network, the large size of the CNN-based detectors, and collecting a large amount of labeled data with enough representation for all objects [36, 118].

Robustness

Robustness in object detection refers to the difficulties of the detector in detecting objects of different appearances. The appearance of the object instance, such as intra-class and inter-class variations, makes the detection task even more challenging. Intra-class variation refers to separate object instances from the same class category. Inter-class variation refers to similar objects from multiple class categories. As shown in Figure 2.8, a chair can be of different color, shape, size, and texture. Also, they are collected with diverse backgrounds, lighting conditions, and view-angles.

Figure 2.8 An example of inter-class variation from the indoor dataset [4]. These are types of chairs captured in diverse conditions such as background, lighting, and viewpoint.

Large Network Size

Recently, the size of CNN-based detectors has been increasing, aiming for more robust detection on challenging tasks. Modern object detection networks consist of billions of parameters, and training of these networks often takes weeks on clusters of high-end GPUs and TPUs. Network speed versus accuracy, commonly known as a latency-accuracy tradeoff, is a widely discussed topic. The rapid growth of lightweight devices such as mobiles, edge devices, embedded and IoT devices, and cloud computing platforms has shifted interest towards network latency. This resulted in more publications on lightweight networks for different object recognition tasks.

Large Amount of Labeled Data

Modern CNN-based object detectors are capable of solving challenging tasks with millions of parameters. However, a huge amount of labeled samples are needed for correctly training detection networks to obtain better results via generalization. The necessity of large-scale diverse and clean datasets to train and validate the object detector has been partly solved by the practice of publicly benchmarking the labeled datasets. However, most of these datasets were collected from a specific environment that might not serve all tasks perfectly. A common approach to solving the data requirements is to apply transfer learning; train a detection network on the relevant public benchmark dataset, and use that network weight to fine-tune the detection network on custom data collected for the specific task. Our publications I, II, and III present an effective way to collect and annotate custom data even with a single human annotator.

2.4 Image Annotation

In a typical setting, training a CNN-based supervised object detector requires a large amount of labeled samples for robust detection performance. The quantity and quality of the labeled dataset play a vital role in supervised learning. However, labeling large enough data with quality labels is labor-intensive and expensive. The rapid growth of the field resulted in various annotation tools and services including Labe-

lImg [1], VGG Image Annotator [19], and Hasty [2]. In a large-scale annotation campaign, crowdsourcing platforms have been used, while in-house tools or services are typically used for small-scale projects. In computer vision, image labeling is primarily categorized into three categories based on the labels assigned; class level, instance level, and pixel level, depending on object recognition tasks.

Annotation for Image Classification

The image classification aims to classify the whole image to a class label based on the content. Unlike object detection, the classification algorithm is interested in only image labels, and one or more suitable labels are assigned to each image. The image annotation for the classification task is easier and faster than other object detection and segmentation tasks.

Annotation for Bounding Box Detection

Bounding box annotation is the most common approach of image labeling for the object detection task. All parts of the object of interest are covered with rectangle boxes keeping the object inside the rectangle. The aim is to draw the rectangle as tight as possible, with all object parts inside it. The box location is typically represented as one corner (top-left or bottom right) and the object's width and height. 2D and 3D bounding boxes have been popular in object detection. While the 2D bounding box is easier to collect, the 3D is more challenging but informative for better visual understanding in challenging scenes.

For image annotation, there are some guidelines available [3] to mitigate annotator biases and ambiguity for quality annotation. Such guidelines can minimize subjectivity ambiguity but they cannot guarantee an absolute perfect case due to unpredictable circumstances such as tiredness of the annotator and external distractions. Summary of the guideline followed in this thesis is presented in Table 2.1.

[1] `https://github.com/tzutalin/labelImg`
[2] `https://hasty.ai/`
[3] `http://host.robots.ox.ac.uk/pascal/VOC/voc2011/guidelines.html`

Table 2.1 Guidelines for annotator for the object detection data annotation

Acceptable objects	All objects within the defined categories minus: - objects that are unclear - objects that are smaller that the threshold (at your discretion) - objects that are heavily occluded (80-90%) - reflection of objects on any surface - a cluster of objects that are hard to separate instances from
Bounding Box	Draw the bounding box around the visible part of the object.

Annotation for Image Segmentation

Image segmentation is considered more effective for extracting the visual context of the image as it is based on pixel-level annotation. Segmentation is further classified into three categories: instance, semantic, and panoptic segmentation. For image segmentation, two conventions are followed: any countable entity is called *thing*, and an uncountable amorphous region or similar appearance is called *stuff*. A visualization of the different image segmentation variants is shown at the bottom row of the Figure 2.3.

Annotation for Instance Segmentation

Instance segmentation is typically done with two major approaches: the top-down approach and the bottom-up approach. In the top-down approach, the bounding box is drawn around the object and then segment the instance mask in each bounding box to distinguish the instance of the object. The top-down approach is also known as the detect-then-segment approach. While in the bottom-up approach, the instance pixels that are close together are combined and kept further away from the pixels of different instances, creating an affinity relationship and grouping similar pixels to outline instances.

Annotation for Semantic Segmentation

Semantic segmentation is a process of assigning every pixel of an image with a class label. In the case of multiple object instances from a single class category, it treats

them as a single entity in the case of multiple object instances. Semantic annotation is comparatively easier than instance annotation as the aim is to label the semantic category of the image.

Annotation for Panoptic Segmentation

Panoptic segmentation unified above mentioned two segmentation techniques to segment images. It semantically distinguishes all objects and identifies separate instances of each category in an image. Among all these methods, panoptic segmentation is considered the most effective technique for better visual understanding as it allows both category-wise and instance-wise image segmentation. The annotation for panoptic segmentation is challenging as it requires both instance (*things*) and semantic (*stuff*) information for each training sample. Unlike other annotation approaches, panoptic segmentation does not support overlapping labels as it assigns a unique semantic label and a unique instance id to each pixel of the image.

2.5 Evaluation Metrics

Unlike classification, object detectors assessment is complex due to the combination of two tasks, classification and localization. The perfect match of bounding box coordinates of the ground truth box and the predicted box is rare. Due to this, object detection performance is measured based on the areas of overlap. The more the predicted bounding box (B_p) overlaps with the ground truth bounding box (B_{gt}), the better the model performance. An intersection over union (IoU) calculation is simply dividing the area of overlap by the area of the union as shown in Eq. 2.1.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \in [0, 1] \tag{2.1}$$

Figure 2.9 demonstrates different scenarios for the predicted bounding box and the ground truth bounding box. The IoU must be greater than the detection threshold value to consider a prediction as correct. In object recognition tasks, 0.5 (IoU @0.5) is commonly used as a threshold value for the correct detection [21, 54, 75].

(a) Excellent (b) Good (c) Poor (d) Off detection

Figure 2.9 Examples of 4 different scenarios of object detection with ground truth label in red and predicted bounding box in green. In (a) the prediction is almost perfect, IoU approaching 1, in (b) the prediction is good, IoU is more than 0.95, in (c) the prediction is not good as IoU is less than 0.5, and in (d) the prediction is way off, IoU reaching 0.

Precision

Precision is the ratio of correctly detected samples among all the detection. In object detection, the precision metric reports the percentage of found objects by the detector that are actual targets. Precision considers all the detections made by the detectors. Precision metric is defined as

$$\text{precision} = \frac{\text{\# of correct detections}}{\text{\# of all detections}}$$

Recall

Recall is the ratio of correctly detected samples among all the relevant targets. The recall is also known as sensitivity or true positive rate. Only the correct detections predicted by the detectors are considered for the recall calculation. Recall metric is defined as

$$\text{recall} = \frac{\text{\# of correct detections}}{\text{\# of all objects}}$$

Average Precision

Average precision (AP) refers to the area under the precision-recall curve. The class-wise AP's is calculated first for each class separately and then average over all classes to produce mean Average Precision (mAP). The mAP metric is widely used as a standard evaluation metric for object recognition and detection tasks. Sensitivity of

the detector can be adjusted using a detection threshold set by default at 0.5.

Mean Absolute Error

The Mean Absolute Error (MAE) metric computes the average error over all predictions. MAE measures the magnitude irrespective of their direction. MAE is more robust to outliers than a mean square error (MSE) as it does not use squaring that amplifies the error. MAE is also known as L1 loss. MAE is used to know how close is the prediction compared to the ground truth; a smaller value is better. We used this metric to evaluate the errors (in years) at the age prediction stage of detected faces in Publication VI.

Accuracy

Accuracy is the fraction of correctly classified instances among all instances. It describes how accurately the model performs on unseen data. The accuracy value lies in the range of 0 to 100; a higher value is considered a better result. Accuracy is a commonly used evaluation metric in machine learning.

FLOPS

Floating-point operations per second (FLOPS) explicitly measures the arithmetic operations involving floating points numbers. We use FLOPS in our experiment to compare the computational complexity of different networks.

Frames per second

Frames per second (FPS) is used to estimate the inference speed and network latency. FPS calculates how many unique still frames are processed in a second. Decent frame processing speed, $FPS > 20$, is crucial for applications aiming for real-time inference. FPS measurement is typically used for a system consisting of user interaction and visualization.

2.6 Workload Calculation

The amount of manual work required to annotate the dataset is based on the number of object instances. In our experiments, the manual workload calculation is based on the number of bounding boxes drawn from scratch, incorrect predictions for locations, and class labels. We did the simulation experiments on how the 2D bounding box annotation could have been done efficiently with the proposed schemes in publications I, II, and III.

Manual Annotations

The workload calculation is based on the amount of manual work required to label each instance *i.e.,* to draw a bounding box for each object. The human annotator labels each object in each image resulting in manual workload same as the number of actual objects given as

$$\#\text{ manual annotations} = \#\text{ of true objects in images.}$$

Annotation Corrections

The correction work on the annotation proposals includes adding new labels and removing unwanted labels. Since recall value represents the proportion of correct detections over all available objects, it complements the required addition value. Thus, the number of required additions can be defined as

$$\#\text{ additions} = (\#\text{ of true objects}) \times (1 - \text{recall}).$$

Similarly, precision value represents the proportion of correct detections among all detections, it complements the required removal value. Thus, the number of required removals can be defined as

$$\#\text{ removals} = (\#\text{ of all detections}) \times (1 - \text{precision}).$$

However, adding or removing a label do not cost the same amount of time. Typically, removing a bounding box is much faster than adding a new annotation from

scratch. Therefore, the workload calculation of the total working time gives a clear perspective.

$$\text{time} = t_s\,(\text{manual annotations}) + t_c\,(\text{additions} + \text{removals}),$$

where t_s denotes the time required for a single object annotation from scratch, and t_c is the average time for a single object in the correction stage. In publication III, we later redefined the total time calculation as

$$\text{time} = t_s\,(\text{manual annotations}) + t_s\,(\text{additions}) + 0.5 \times t_s(\text{removals}),$$

Annotation Time

We aim to draw the bounding box as tight as possible while avoiding overlaps between boxes. Drawing a bounding box annotation in an image with a single object takes less time than drawing in an image with many objects. We reported that drawing a bounding box with a class label takes 10.15 *seconds* on average for the Indoor dataset, while simply removing takes less than 5.20 *seconds* in Publication I. Similar results have been reported in other works mentioning that correcting false negative takes twice as long to correct false positive [104].

2.7 Datasets

This section describes the datasets used in the thesis and publications. A summary of the datasets used is presented in Table 2.2.

Table 2.2 Summary of the datasets used in this thesis. Object instances mentioned here are only 2D bounding boxes from the datasets. The datasets used for other tasks are not included in this table.

Dataset	Source	#Images	#Instances	Usage
FDDB	Faces in the Wild	2845	5171	Face detection
KITTI	Street View	1500	80000	Multi-purpose
Pascal VOC	Collected online	17125	40000	Multi-purpose
OpenImages V4	Collected online	9 M	14.6 M	Multi-purpose
Indoor*	Indoor scenes	2213	4500	Object detection

* created by the author, see Publication I for more details.

FDDB

Face Detection Data Set and Benchmark (FDDB) [33] is a face detection dataset containing human face regions. The dataset is designed to study the problem of unconstrained face detection. It contains 2845 images taken from the Faces in the Wild dataset with the annotations of 5171 faces. The FDDB dataset contains both gray scale and color facial images with a wide range of occlusions, different poses, and out-of-focus faces.

KITTI

KITTI [25] is a challenging real-world computer vision benchmarks dataset. This dataset includes 7481 training images and 7518 test images, consisting of 80256 labeled objects. The dataset aims to enhance a wide range of computer vision research; it is collected from five scenarios: city, residential, road, campus, and person. We use the train split from the object detection category consisting of 7481 images with 40570 object instances of 8 class categories, including vehicles and person involved in different actions.

Pascal VOC

Pascal VOC [21, 22] is one of the popular datasets widely used in the computer vision research community [23, 63, 64, 65, 75]. Pascal VOC is the pioneer dataset contributing to the rapid growth of object detection research by organizing the yearly competitions, Pascal Visual Object Classes Challenge, from 2005 to 2012. Over the years, the dataset has been extended to include other vision tasks. Pascal VOC 2012 has annotations for object detection, semantic segmentation, and action recognition. The first version started in 2005 with 4 class categories and extended to the current version of 20 class categories in 2007. Since then, it consists of fully labeled in 20 object classes, including animals, vehicles, and everyday household objects. We used all data from 2007 and 2012 versions, 17125 images with 40K object instances, in publications II and IV. Publication III uses all 9963 images with 30638 object instances from train validation (trainval) and test sets of the 2007 version. We performed two sets of experiments in this dataset: the first sets with all class categories and later with individual class categories.

OpenImages

OpenImages [44] is a relatively new large and challenging dataset for multiple computer vision tasks. It contains image-level annotations, object bounding boxes, object segmentation, visual relationships, and localized narratives. In this thesis, we used OpenImages V4, which contains 9M images in total. The object detection training set of V4 includes 14.6M bounding boxes in 1.7M images with 600 class categories. On our experiments in Publication II and V, we used 10.5K images downloaded from the OpenImages dataset person class, excluding the occluded, truncated, and groups of objects tags *i.e.,* multiple objects inside a single bounding box. The latest version, OpenImages V6 [4], contains 16M bounding boxes on 1.9M images.

Indoor

Indoor dataset [4] is easy to use, small-sized dataset we collected from the university indoor environment. The dataset consists of various backgrounds, lighting conditions, high inter-class variances, scales, and occlusions. The dataset has 2213 im-

[4] `https://storage.googleapis.com/openimages/web/index.html`

ages and about 4595 object instances of 7 indoor scene classes extracted from a series of HD videos preserving temporal order. The dataset consists of safety signs (exit, fire extinguisher), furniture(trash bin, chair), and equipment (clock, printer, screen). The number of an object per class category varies; popular class is chair class with more than 1600 instances, and the least popular is printer class with less than 100 instances.

Other Datasets

In addition to the Face detection dataset, FDDB, the following datasets are used in Publication VI. The Annotated Facial Landmarks in the Wild (AFLW) [58] is used to train the facial landmark detection network. The ChaLearn Looking At People (ChaLearn LAP) [20] is used on training the age estimation network. The GENKI-4K [61] is used to train the emotion recognition network *i.e.*, detecting smiles and non-smiles from the detected face. Finally, the CelebFaces Attributes (CelebA) [55] is used to train the similarity search network; to map the detected faces to similar-looking celebrity faces.

3 COMPUTER ASSISTED IMAGE ANNOTATION FOR OBJECT DETECTION

This chapter summarizes our works on efficient annotation approaches for the quality training dataset. The challenge in getting the quality annotation for object detector training and the impact of label noise on detector performance is discussed shortly. We present our annotator-friendly machine-assisted approach for the faster, less tedious, and cost-effective 2D bounding box annotation scheme for collecting quality training datasets.

3.1 Label Noise in Object Detection

Object detection network training requires a large amount of example images that are typically labeled by human annotators manually. As mentioned in Section 2.4, drawing bounding boxes on image datasets is tedious, costly, and error-prone. In [83], authors report the average time of 42 seconds to draw and verify a single quality bounding box for the ImageNet dataset, a large-scale annotation campaign, by crowdsourcing on AMT. Moreover, additional time is required to prepare annotation guidelines, train the annotators, and manually approve the annotation. Due to the enormous requirement of time and resources, crowdsourcing services have been typically used for the large-scale image annotation [22, 44, 51, 78]. Recently, automated data generation has been proposed for collecting training datasets for different settings [43, 91, 113]. Though crowdsourcing and auto labeling ease the data collection for supervised learning, they often cause random labeling errors.

In a typical setting of the data annotation project, annotation involves soft skills that usually do not require expert knowledge of ML engineers and data scientists. Annotation is often done through practical experiences rather than formal training. An instruction of predefined rules is provided to the annotator for the task. How-

ever, even with the predefined rules, the annotation of large-scale datasets with many human annotators is prone to noise. We consider two factors for the label noise: (1) the involvement of many annotators from diverse backgrounds in the crowdsourcing annotation leads to subjective bias from the human annotators, and (2) the tedious and challenging annotation task in limited time naturally adds random noise as the annotator tends to skip or mislabel some data.

A machine learning model is only as good as the data it is trained on. Highly imbalanced datasets, small-sized datasets, noisy datasets, and poorly sampled datasets are often cause of the poor generalization of machine learning models [1]. The quality of data is crucial as poorly labeled training data do not reflect the real-world circumstances and will likely result in a machine learning system that is ill-equipped to operate. The data quality requirement is even tighter in the case of modern deep learning algorithms, where tuning hundreds of millions of parameters requires a large number of quality samples to get a good representation model.

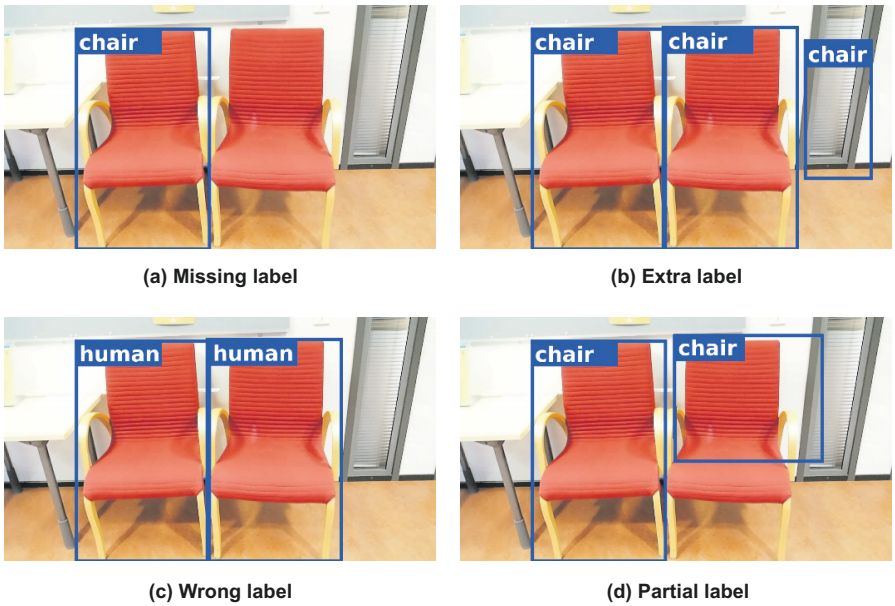
In publication IV, we aim to study the sensitivity of object detection loss functions towards label noise. To this end, we first categorize types of possible noise scenarios in the object detection dataset as shown in Figure 3.1. Missing labels is probably the most common type of noise in object detection, which was simulated in this paper by removing random annotations from training dataset to explore the effect on two different loss functions; cross-entropy and focal loss. We experimented with varying amounts of noise in the datasets with a one-stage object detection network, SSD head with MobileNet V1 backbone, and with different focal loss hyperparameter values, $0, \ldots, 8$. Our extensive experiments demonstrated that focal loss suffers more when facing noisy labels. However, it can be mitigated by careful selection of hyperparameters, namely the gamma (γ) value.

Related Works

Researchers have proposed a list of different types of noise for the image classification task while studying the effect of each type [24]. Based on this work, we defined the most common types of label noise in object detection as shown in Figure 3.1.

BundleNet utilizes sample correlation technique by separating samples based on

[1] `https://hbr.org/2018/04/if-your-data-is-bad-your-machine-learning-tools-are-useless`



(a) Missing label (b) Extra label

(c) Wrong label (d) Partial label

Figure 3.1 Common types of label noise in object detection: (a) missing label, the other chair is not labeled, (b) incorrect annotation, part of the window on the chair, (c) wrong classification label, humans instead of chairs, and (d) partial label, inaccurately drawn bounding box resulting in low IoU. Used with the permission from publication IV.

their class and treating them as independent inputs to improve the robustness of the network against label noise [47]. CleanNet detects noise in the dataset, and provides a clean dataset which can be utilized by another classifier for better performance [45]. MentorNet uses a mentor network to supervise the training of the base network called StudentNet for training deep CNNs on corrupted labels [35].

The robustness of two-stage object detectors under the presence of missing annotations has been studied in [101]. The method uses soft sampling to re-weight the gradients of RoI based on the overlaps with positive instances that provide smaller weight to the uncertain background regions than hard negatives. The background re-calibration loss method is inspired by the focal loss to handle the missing annotation for the one-stage object detectors [110]. This method can automatically re-calibrate the loss function based on the pre-defined IoU threshold and the input image.

Label noise is also considered as one of the critical factors for the safety of deep learning algorithms [96, 98]. Zhang *et al.* review problems related to the dataset, such as label noise, by surveying over recent works with the suggestion to use a robust loss function and re-weighting samples to mitigate this issue [112]. Researchers

have suggested implementing a proper labeling guideline to mitigate the effect of label noise. However, the process of manually labeling large datasets will always be prone to noise due to factors such as multiple human involvements, variations in data, and the tedious nature of the task.

Recently, there have been increasing publications resolving label noise problems with the newer type of training procedures and loss functions [14, 68]. Label noise in ML training is gaining significant interest from the researchers resulting in publications covering different perspectives to mitigate the issue [2].

Results and Discussion

Our experiments used 5 different noise levels: 10%, 20%, 30%, 40% or 50%, of missing labels on training two loss functions for object detector training; cross-entropy and focal loss.

The training datasets were initialized by randomly removing incremental amount of labels from the dataset. Both networks were trained on the same dataset to keep the comparison fair. Both models were trained simultaneously with a gamma value of $\gamma = 2$ for focal loss as proposed in [50]. The *mAP@.50 IoU* (mean average precision with 0.5 IoU threshold) is used as a performance evaluation metric. As illustrated in Figure 3.2, for all datasets, the detector with focal loss performs worse on all noise levels.

Next, we experimented with 0%, 10% and 50% of label noise with different values for the focal loss gamma parameter, $\gamma = 0, 1, 2, \ldots, 8$. As illustrated in Figure 3.3, for all three datasets, a higher value of the gamma tends to perform better for the noisier dataset. Our experiments with a one-stage object detector with two loss functions demonstrate that focal loss is more sensitive to the label noise. We also observed that the robustness of the detector can be improved with minimal effort by adopting a higher value of gamma (γ) parameter.

In contrast to the Wu *et al.* findings that suggested deep learning-based object detectors are relatively robust to missing annotations [101], we demonstrated that one-stage deep learning-based object detectors are sensitive to missing annotation. However, our result suggested that for a more noisy level, using focal loss with larger γ values can improve the robustness of the detector. In these cases, we believe that

[2] `https://github.com/subeeshvasu/Awesome-Learning-with-Label-Noise`

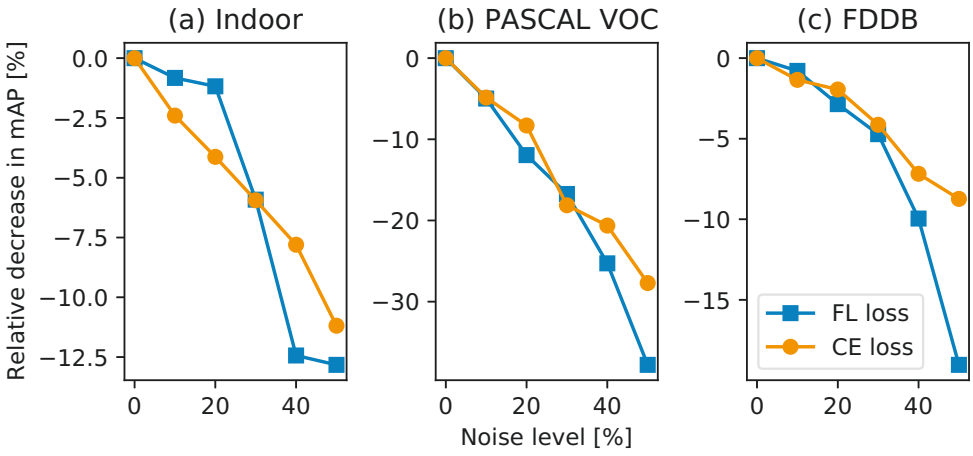


Figure 3.2 Relative decrease in object detection performance (mAP) with different amount of label noise on three datasets.

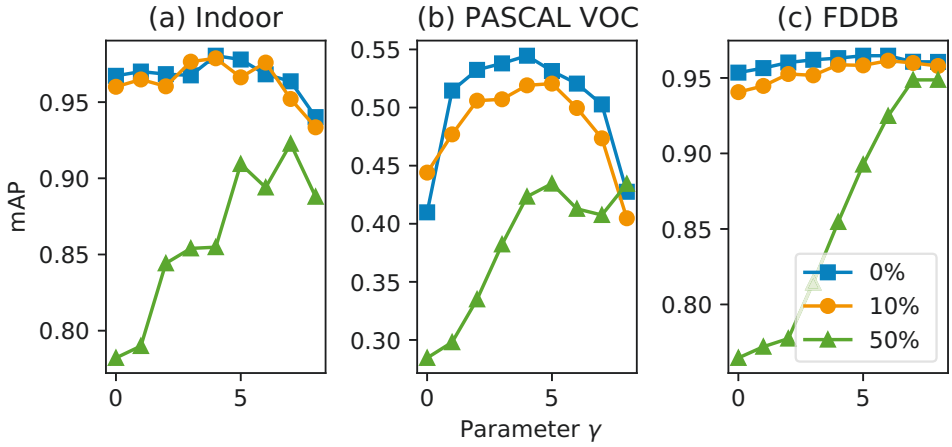


Figure 3.3 Results on three datasets with different gamma values on 0%, 10% and 50% noise levels.

the model focuses on the complex samples for training, *i.e.*, the annotated samples with low confidence and non-annotated samples with high confidence.

3.2 Two-stage Approach for Bounding Box Annotation

In Section 3.1 we discussed the issue of label noise in training deep learning-based object detectors. We believe the involvement of multiple annotators from diverse backgrounds in the crowdsourcing annotation and labeling many images at a single pass is a major cause of noise on the data annotation. In this section, we proposed annotator-friendly methods to mitigate the possible noise by utilizing the information from the trained detectors. To this end, we propose a two-stage bounding box annotation method where annotators simply annotate a subset of unlabeled images manually and utilize the knowledge learned from the object detection network trained on that subset to label the remaining unlabeled subsets.

As data annotation is one of the challenging factors for deep learning, automating the image annotation helps to reduce the time and cost constraints to implement deep neural network models in computer vision problems. Moreover, our method utilizes the concept of human-machine collaboration for faster bounding box annotation to train deep learning-based object detectors.

In recent years, the typical workflow in deep learning projects is to apply transfer learning [86] where the knowledge learned from a generic task is transferred to train a model for other specific tasks. In a typical setting, the network is first trained on a large publicly available generic dataset such as COCO, Kitti, and OpenImages. Then, the weights are finetuned on a smaller set of domain-specific data to enhance the performance of the network.

There are a large number of tools and services available to perform data collection and labeling for ML algorithms training. However, working with these tools and third-party services is challenging; data quality is often an issue with high latency and operation costs. In contrast, there is much more freedom and control when using in-house data labeling services. Data quality can be maintained, the constant update is relatively easy and cost-effective, and data privacy is owned with the cost of setting up data annotation teams. For example, enormous data labeling for Tesla slowed down their development process [3], so they opted for in-house data labeling with more than 1000 human annotators.

[3] `https://www.tesla.com/AI`

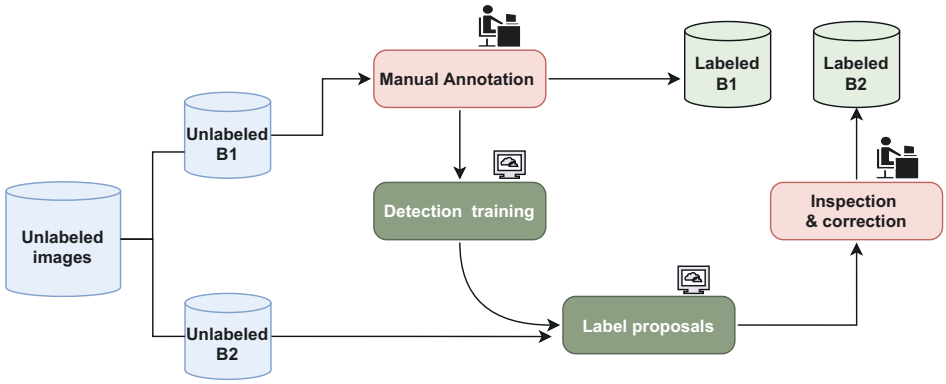


Figure 3.4 Two-stage train annotate approach for faster bounding box annotation. Reproduced with the permission from publication I.

Related Works

In recent years there has been growing interest in studies focusing on how to speed up the image dataset annotation [40, 63, 64, 65, 66, 78, 83].

Su *et al.* [83] focused on the crowd-sourced annotation process in object detection. They divided the process into into three sub-tasks: (1) drawing the boxes, (2) verifying the quality of boxes, and (3) verifying the coverage of boxes on a single image.

Papadopoulos *et al.* proposed multiple approaches for efficient bounding box annotation. In Eye-tracking scheme [63], they track the human annotator eye movements to extract the information of the size and position of the target object. In the bounding box verification scheme [64], they use human annotators to verify the label proposed by the network with an accept/reject decision. In the click supervision scheme [66], the human annotator is requested to mark the point in the tentative center point of an object in an image. While in the extreme clicking scheme [65], the annotator is requested to mark four extreme physical boundary points on the object: the top, bottom, left, and right-most points.

Russakovsky *et al.* [78] proposed a method to integrate multiple humans with computer vision for collecting large-scale datasets with minimal supervision. Intelligent Annotation Dialogs by Konyushkova *et al.* [40] utilizes an automatically selected action sequence (box verification and manual drawing of box) for a human annotator to generate high-quality bounding boxes.

Method

The components of our two-stage annotation workflow are illustrated in Figure 3.4. The first step in our workflow is to splits the dataset into two subsets. We experimented with the folds of 1% to 10% with 1% increases and 15% to 95% with 5% increases for the first subset *B1*. The corresponding second subset *B2* for each *B1* is always the remaining percentage, so a total of 100 % of the data was used in the experiment.

The second step is to request the human annotator to fully annotate all images from the unlabeled subset *B1*. As the task is 2D object detection, a tight bounding box with an object class label is drawn for each object instance in each image.

The third step is to train the object detector using the pretrained weight on the recently labeled subset. Although any detector can be used, we experimented with popular one-stage and two-stage detection networks. The detection network is finetuned on manually labeled *B1* using the pretrained weight from the COCO dataset. More specifically, we trained on the Faster RCNN model with ResNet-101 backbone network trained on the MS COCO dataset as our starting point.

After finetuning the object detection model, it is then used to predicts bounding boxes and class labels for the rest of the unlabeled images in subset *B2*. Finally, the proposed predictions are inspected and corrected by the annotator. The human is asked to go through all the proposed labels, check and correct if necessary.

Results and Discussion

Figure 3.5(top) shows the relation between the required workload for fully annotating the dataset versus the amount of data used in *B1*. Note that the required workload is below the estimated time to fully annotate the dataset without using the proposed method in all cases. Judging by the results, there is a trade-off between the workload of annotating data in *B1* and the workload required to correct the proposed annotations in *B2*. The results suggest that a good point lies in a relatively low percentage for *B1*, around 4 – 8%, which causes the whole annotation process to be 9× faster compared to the fully manual annotation on the Indoor data.

In our experiment with the Indoor dataset, we found out that, on average, manual annotation takes about 10.15 *seconds* per bounding box, and correction (addition

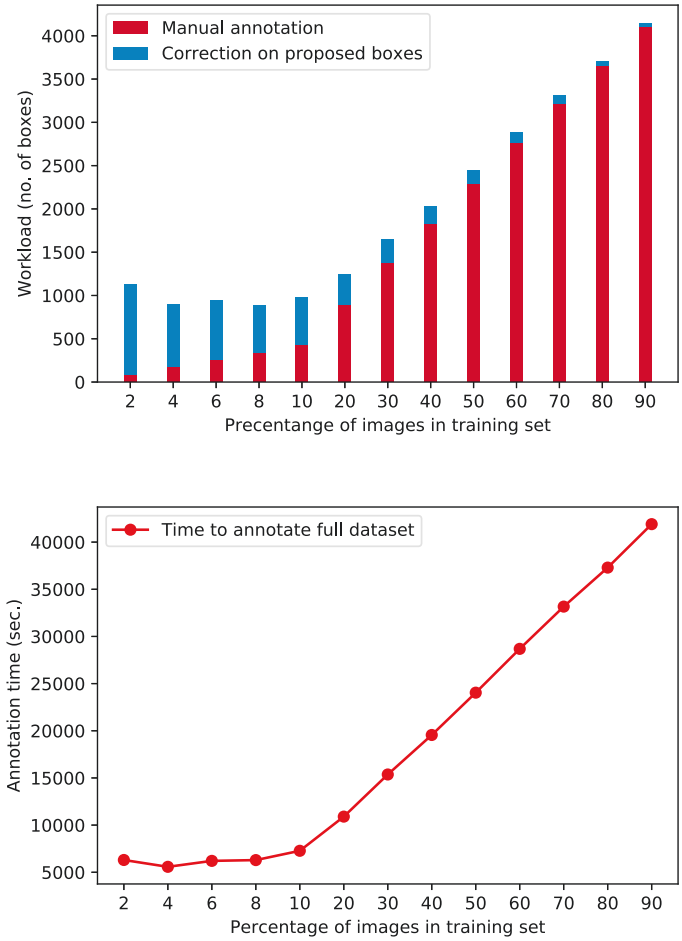


Figure 3.5 Amount of workload needed in different train-test split (top). The relation between the required time for annotation versus the amount of manually labeled data in training set (bottom). Used with the permission from publication I

and removal) takes about 5.20 *seconds* per bounding box. More details on the experiment and the calculation method can be found in Publication I. We consider this number for the calculation of total annotation time depreciated in the graph. The total workload required by the proposed method is upto 90% less than manually labeling all images in the dataset. Keep in mind that the amount of workload might differ for other datasets. As illustrated in Figure 3.5, knowing the perfect split for the minimal human annotation efforts is of interest for the data annotation campaign.

In addition to the faster bounding box annotation approach, we publish the fully labeled Indoor dataset for registration-free use for the research community. We believe this dataset can be helpful for the quick experiment on training object detectors as the training from scratch can be done in a few hours with decent computing resources.

3.3 Iterative Approach for Bounding Box Annotation

In Section 3.2 we proposed a method for labeling object detection datasets by using the trained detector on the subset of the manually labeled data. Extensive experiments demonstrated the effectiveness of our approach, and it managed to save a large amount of manual annotation effort. However, as discussed above, the drawback of the proposed method is that saving manual annotation workload varies significantly depending on the size of the two splits created prior to the manual labeling, thus making it unsuitable for all types of data annotation campaigns. This section presents our efficient annotation method that uses a small amount of images over many batches and utilizes the human-in-the-loop principle to create bounding box annotation efficiently. Moreover, this method overcomes the disadvantage of the two-stage method as it eliminates the need to carefully sample images over two subsets, B_1 and B_2.

Similar to the aforementioned two-stage annotation approach, the iterative annotation approach utilizes the human-machine collaboration for efficient bounding box annotation. The main difference is that the iterative human-in-the-loop annotation approach does the training, label proposals, and correction in an iterative loop until it reaches the desired performance or labels all data.

Related Works

Human-Machine collaboration for image Annotation have been actively researched for the efficient data labeling for machine learning training [11, 57, 78, 100]. Russakovsky *et al.* [78] proposed a human-in-the-loop annotation method, where the objects in the image is detected using state-of-the-art networks. Afterwards human annotators are asked to detect any missing object on the same image. While it is a time-consuming task, due to the requirement of finding all objects in every image it was necessary.

Lutnick *et al.* [56] proposed an iterative human-AI loop for efficient semantic labeling on medical images. Their idea is similar to our proposed iterative annotation scheme but applied for semantic annotation on medical images.

Other works on efficient image annotation using humans-in-the-loop includes semantic annotation of objects in image and video datasets [1, 9, 52]. Polygon RNN [12] and Polygon RNN++ [1] use recurrent neural network architecture to produce polygonal annotations for objects which is then corrected interactively by human annotator. Curve-GCN [52] uses Graph Convolutions Network (GCN) to predict all vertices of the object simultaneously. Berg *et al.* [9] suggested using a semi-automatic method to annotate video data. This method utilizes a state-of-the-art object segmentation to propose initial annotations for all frames in the video.

Method

The main components of our iterative annotation human-in-the-loop workflow are illustrated in Figure 3.6. The proposed method uses an incremental learning approach. It starts by manually labeling a small batch of images, then trains a detection model with the data, uses the model to propose bounding boxes for the next batch of unlabeled images, requests the human annotator to inspect and correct the proposed labels, uses all the labeled data to train a new detector, and continues. This process continues in a loop until all images are labeled, or the annotation budget is finished. Here the involvement of human annotators is only labeling the first batch and correcting the proposed bounding boxes. Algorithm 1 summarizes all steps of the iterative training method. Next, we will describe each component of our approach.

The first step in our iterative annotation workflow is to split the dataset into small subsets, $B_0, \ldots, B_M$. Then similar to our two-stage method mentioned above, fully annotate all images in the first batch B_0. After that, we train the object detector on recently labeled B_0. To get better performance for object detector, transfer learning techniques with pretrained weights can be utilized. For extensive experiments, we use both one-stage and two-stage detectors. Moreover, we trained on the SSD with the MobileNet V1 backbone and faster RCNN with the ResNet50 backbone. The next step is to use the trained detector to predict bounding boxes and class labels for the next batch B_1. Then, the human annotator will inspect the label proposals and

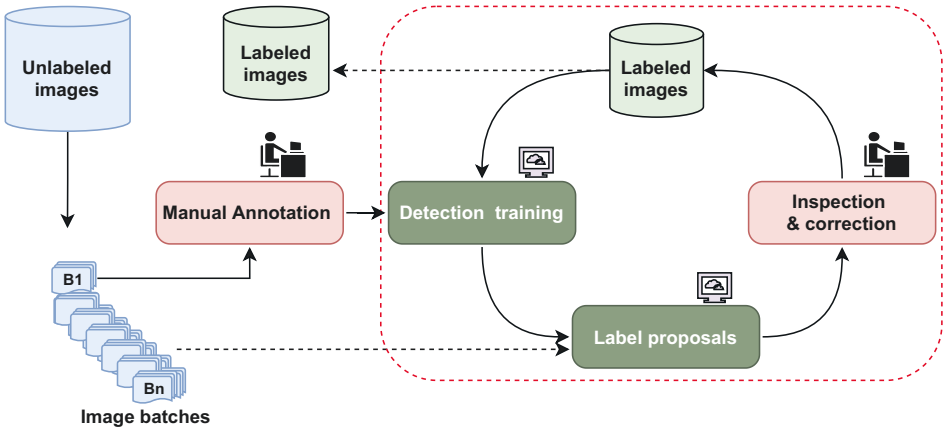


Figure 3.6 Overview of our human machine collaborative approach for the iterative train-annotate approach (publication II). Human annotator and machine works together to create object detection dataset. Reproduced with the permission from publication II

Algorithm 1: Iterative annotation

Require: Set of unlabeled images split to $M+1$ distinct batches $B_0, \ldots, B_M$

1: annotate images on batch B_0 manually
2: train object detector with images from B_0
3: **for** $i \in 1, 2, \ldots, M$ **do**
4: propose labels for batch B_i using the current prediction model
5: do manual inspection and correction for the proposals
6: fine-tune the object detection model with batch B_i
7: **end for**

return fully labeled dataset

corrects them if required. Next, the recently labeled batch B_1 is used to finetune the object detector. Repeating the previous steps for B_2 to B_M the proposed method will result in a fully labeled dataset. It is also possible to stop the loop when either the annotation budget is finished or the desired detector performance is achieved.

Results and Discussion

We proposed an iterative human-in-the-loop annotation approach to minimize manual labor for image annotation. The proposed method can produce high-quality bounding box annotations on both small and medium-sized datasets. Empirical experiments on three different datasets shows the effectiveness of the proposed method

Table 3.1 Reduction of manual workload (%) with different strategies on Indoor, Pascal VOC 2012 and OpenImages V4 (Person class) datasets.

Network - Approach	Indoor	Pascal VOC	OpenImages
RCNN - Shuffled	**75.86**	18.40	45.62
RCNN - Sorted	56.97	20.93	**60.05**
RCNN - Original	35.78	**25.23**	45.73
SSD - Shuffled	**47.38**	3.46	20.28
SSD - Sorted	31.58	5.66	**35.13**
SSD - Original	19.24	**7.97**	20.04

in reducing the manual workload by up to 75%. Based on the experiments, the dataset size, image source, and object class categories seem to affect the amount of workload reduction. Moreover, the proposed method managed to perform adequately on the existing annotation platforms.

Unlike the two-stage approach, the nature of the proposed method makes it easier to split the first batch of data as there is no concern for choosing the correct hyperparameters. Also, the detector gets more accurate over the course of labeling, making it progressively easier for the human annotator to inspect and correct the proposals.

However, from our experiments, we noticed that the reduction in annotation workload heavily relies on how the images are chosen for the first few batches. In our simulated experiment, we utilized the ground truth label information, such as the number of objects in an image. As our dataset is fully labeled, obtaining such information was relatively easy, but there might be no information available about the unlabeled images in a real-world project.

3.4 Sample Selection for Efficient Object Annotation

As previously mentioned in Section 3.3, we proposed a human-in-the-loop iterative annotation method to reduce the workload for human annotators. However, in our experiments, we found the sensitivity of the proposed method to the algorithm used for sorting the data into batches. In this paper, we studied multiple approaches for actively selecting image samples for each batch to enhance the quality of our proposed method. The significant differences of our work compared to others are: (1) it does not require explicit information from the unlabeled images, (2) all the sampling task is done prior to the training phase, and (3) the entire image is used to extract features.

Related Works

Utilizing sample selection techniques have proven to be effective in reducing the cost of annotation in different tasks [18, 49, 105]. These methods works by iteratively training a model with the labeled samples and finding the most informative unlabeled samples to be labeled for the next batch. While the uncertainty-based metrics for finding the informative samples shown promising results, they require a labeled dataset for building the initial model, which is impractical in real-world cases where the amount of labeled samples are limited.

Smailagic *et al.* [82] proposed active learning (AL) methods to create an optimized labeled training set from unlabeled medical images. This method utilized the features generated by a feature descriptor function to select the unlabeled samples with most euclidean distance from the labeled samples. The drawback of this method is the high computational cost on multi-class datasets.

Recently, self-training approaches have been proposed for image annotation where a trained model predicts labels for a set of unlabeled images and involves humans in the process of correcting predicted labels [40, 56].

Wong *et al.* utilized contextual sampling criteria to propose an assistive learning feedback loop. In this method, the samples are selected based on both the uniqueness and the average euclidean distance [97]. However, such information is not typically available for real-world datasets.

Method

The components of our sample selection workflow are illustrated in Figure 3.7. First, a CNN network is used to extract features from all images, which is then used to compute the pairwise euclidean distance of the entire dataset, stored in $N \times N$ matrix (N is the number of images in the dataset). Unlabeled images are then sampled in mini-batches based on two sorting schemes: (1) small distances together, *i.e.,* the most similar images compared to all previously selected images are kept in a batch and (2) large distances together, *i.e.,* the most distinct images compared to all previously selected images are kept in a batch.

We also created a custom similarity network for the feature extraction. The similarity network is based on the ResNet50 backbone trained on pairs of images horizontally divided in half using triplet loss [80] as the loss function.

The human annotation effort is computed in terms of time (seconds) and annotation effort calculation in terms of bounding boxes. As the addition and removal task complexity differ in real annotation cases, we assumed that adding one bounding box and class label during the correction stage is as costly as doing the complete manual annotation done in the first batch of unlabeled images.

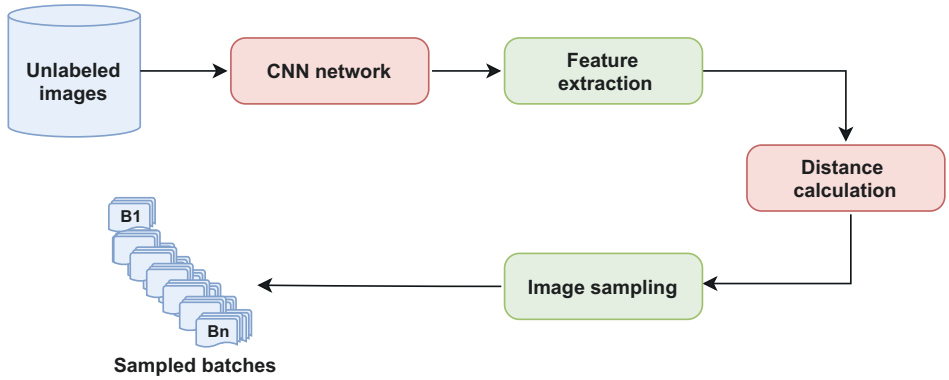


Figure 3.7 Overview of our sample selection for object detection data annotation approach. This approach is used in publication III with our iterative annotation approach proposed in publication II. Reproduced with the permission from publication III

Table 3.2 Reduction of manual annotation time W_T (%) (higher the better) with different approaches. The result is an average of five independent runs.

Dataset	Approach	RCNN		SSD
		0.5 IoU	0.7 IoU	0.5 IoU
Pascal VOC	Shuffle	56.00	55.37	31.61
	Sim (ImgNet)	56.05	54.71	**32.97**
	Dis (ImgNet)	56.48	55.63	32.08
	Sim (SimNet)	55.27	53.65	30.53
	Dis (SimNet)	**56.82**	**56.02**	32.08
KITTI	Shuffle	50.82	49.53	32.21
	Sim (ImgNet)	37.31	34.68	19.92
	Dis (ImgNet)	**51.49**	**50.59**	**33.61**
	Sim (SimNet)	46.67	43.34	28.64
	Dis (SimNet)	49.81	47.16	28.89
Indoor	Shuffle	81.20	80.37	65.98
	Temporal	37.47	35.99	20.24
	Sim (ImgNet)	59.12	59.47	43.35
	Dis (ImgNet)	**81.83**	78.81	64.09
	Sim (SimNet)	67.06	67.38	48.32
	Dis (SimNet)	79.08	**81.64**	**67.76**

Results and Discussion

Image feature extraction is the first task of our proposed system. We used two CNN networks for the feature extraction, ResNet50 trained on the ImageNet dataset and a custom-built similarity network (SimNet) trained on each dataset to be labeled. We used three schemes to sample images over batches: similar, dissimilar, and random shuffle. In addition, we used temporal order if applicable to the dataset. We experimented with SSD and faster RCNN detectors with a 0.5 IoU threshold value for the correct detection. Some additional experiments were done with a 0.7 IoU value.

As shown in Table 3.1 the annotation workload for both Indoor and Pascal VOC datasets have reduced with the sample selection approach presented in publication III. As shown in the table, the reduction in the workload is higher with an IoU

value of 0.5 compared to the IoU value of 0.7 since the network will predict fewer bounding boxes with the higher IoU threshold, thus requiring the human annotator to draw more bounding boxes from scratch.

There are also other time-consuming tasks in the pipeline of the proposed approaches, such as the time required by the machine to extract the features, calculate the Euclidean distances, select the samples, and train the detection network. However, these are outside of the scope of this work since the machine does not require human intervention in these tasks.

Alternative Methods for Efficient Object Detection Training

In this thesis, we aimed for an efficient bounding box annotation approach for the moderate size data annotation campaign for training deep learning-based supervised object detectors that can be handled with fewer human annotators. Other alternative methods for efficient object detection training include other types of object detectors that do not require a large amount of the labeled dataset to train and alternative approaches to collect training datasets.

Unsupervised [15, 16], semi-supervised [34, 115], and weakly supervised [10, 37, 116] object detection approaches can reduce the data annotation burden with considerable detection accuracy utilizing sparse annotation [118].

On the other hand, weak annotation techniques are helpful to create a high-quality training dataset with a modest annotation effort for a decent detection model [65, 66]. Synthetic data generation is another helpful technique for generating noise-free training datasets with less human annotation efforts [48, 113].

4 EFFICIENT IMPLEMENTATION OF OBJECT DETECTORS

Machine learning applications have been rapidly increasing in various fields such as agriculture, face analysis, medical imaging, video surveillance, and robotics. While the current state-of-the-art object detection models achieve high performance, their computation and memory requirements are often the limiting factors for the real-time implementation on resource-limited devices.

The accuracy of CNN-based detectors is directly related to the computational resources available to them. Meanwhile, embedded devices are becoming popular in commercial and industrial settings, allowing machine learning systems to thrive in broader real-time projects and applications. Due to the limitation of resources in these devices, efficient implementation of object detectors is essential for the further expansion of these applications.

We designed and deployed object detection networks on imaging pipelines while achieving decent detection speed for different applications. In particular, a face detection network is integrated into a facial analysis system, a person detection network is deployed in a privacy-aware person tracking, and a custom object detection network for a trajectory prediction system to detect moving objects. Moreover, we studied the practical implementation of distributed computer vision systems adopting deep neural object detectors for resource-limited devices.

Object Detection Development Workflow

The typical workflow in the deep learning object detection application is depicted in Figure 4.1, details of each stage are as follows: The first step is defining the task and deciding on the frameworks: Purpose of system targets for the detector, environment of usage, limitation of devices, and platform or a framework of the system.

The second step is to collect and create datasets. Based on the task and the framework, the collection and labeling of the dataset are done.

The third step is to start training the network either from scratch or by using pretrained weight. To assess the quality of training, validation set can be used. The training step is usually repeated multiple times to tune the hyperparameters till a satisfying result is produced.

Finally, the best-performing model is chosen for testing and inference. Since the deployment criterion varies depending on the target device, the model is frozen accordingly to meet the requirements.

Additionally, steps two to four must be repeated to expand or update the model with new data, labels, features, etc.

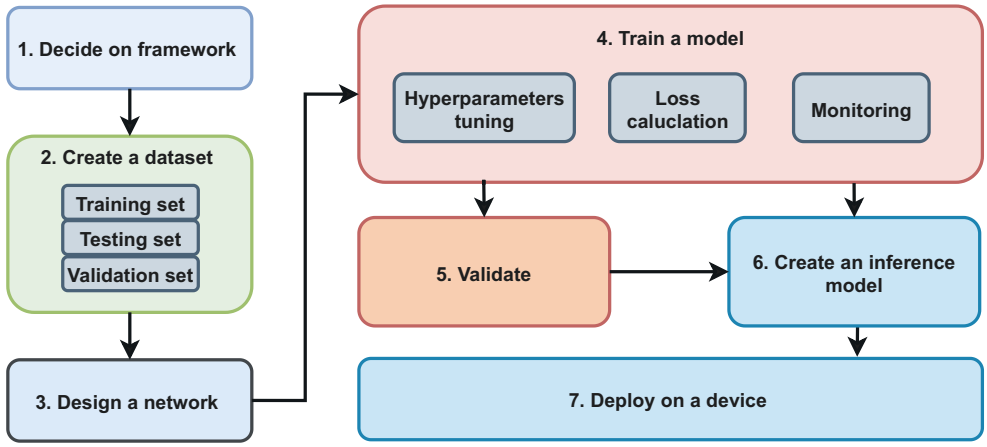


Figure 4.1 Typical workflow of deep learning-based object detection projects.

Object Detection on Low Resourced Devices

Object detection serves as a core part of the system for a wide range of applications that aims to understand and recognize the visual context of the surrounding environment. In a typical setup, modern object detectors cannot have real-time inference without GPU. However, in some circumstances, heavy computing devices are not suitable for deployment due to the nature of the task and the location where the system is installed. For example, surveillance systems on the street, animal movement trackers in a remote park, and obstacle detectors on moving robots cannot use heavy and powerful machines. The single-board embedded device such as Raspberry

Pi, Jetson Nano, and Google Coral AI are popular choices for those scenarios, which require a detector that can perform decently on these embedded devices. Thus, researchers have a long-standing interest in efficient detection networks [32, 60, 95, 99, 102].

The common approaches utilized for the efficient object detection for the low-resourced devices are:

- using an efficient combination of detector head and backbone network. For example, the inference speed of SSD light with MobileNet V2 is higher than the SSD with MobileNet V1.
- reducing the dimension of the training data. For example, reducing the input image size for the network increases inference speed without compromising detection accuracy as much.
- using a network pruning approach. For example, using a smaller depth multiplier ($\alpha < 1$) for the MobileNet reduces the computation with a slight decrease in overall detection accuracy.

4.1 Privacy-aware Person Detection System in Edge Device

The object detection system is widely used in the surveillance field. In human detection, data privacy has become a primary issue as there is a growing concern about the ethics of AI. The concerns about collecting personal data, such as images and videos, storing and handling them safely and securely added more challenges to using a system that deals with these data. We aim to provide a privacy-preserving system for the person counting on the premises, such as libraries, museums, and exhibition centers.

Deployment of CNN-based detectors on single board embedded systems is not straightforward due to the heavy computation involved in these networks and the lack of powerful computing resources on the edge devices. This work aims to deploy a multitask computer vision system for person detection, re-identification, and tracking on a cluster of embedded devices aiming for smooth operations with added neural network accelerators, Neural Compute Stick (NCS). To this end, we experimented with multiple design perspectives and combinations of software and hardware to collect only the feature matrices of the people in the field of view and distribute them to the cloud server for further processing.

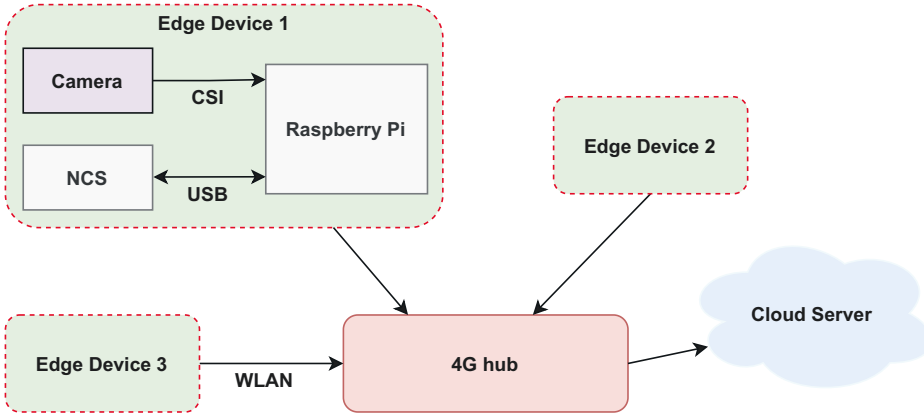


Figure 4.2 Overview of our person tracking system (publication V). Each edge device consists of a camera module, a Raspberry PI, and a NCS. Collected data are sent to the cloud server via WLAN and cellular connections. Reproduced with the permission from publication VI

We presented a feasibility study of simultaneous object detection and feature generation with privacy-aware person tracking on the embedded devices. Our system is deployed on a single board Raspberry Pi with a NCS, as depicted in the system overview in Figure 4.2.

The contributions of this work are as follows:

- We present a privacy-aware person tracking system for cluster of edge devices.
- We study the design prospective for the optimal performance while maintaining the smooth and safer operation.

We used models with pretrained weights on the COCO dataset and fine-tuned them on person class from the OpenImage dataset. Multiple tests were conducted to compare and analyze different network configurations. The configurations are input image size (300x300, 200x200 and 100x100 pixels) and depth multiplier value (α = 1, 0.75, 0.5, 0.25). Figure 4.3 shows the accuracy versus speed for these configurations running on NCS. As shown in the figure, the configuration settings provide a large selection of accuracy-speed trade-off options. Moreover, the results indicate the small variation between different network architectures, with MobileNetV1 being the fastest option. All tests were conducted on two settings for CPU: performance mode (1400 MHz) and power-save mode (600 MHz).

We study the design spectrum for choosing optimal architecture for desired tasks while maintaining smooth and safe performance. Moreover, the results show the

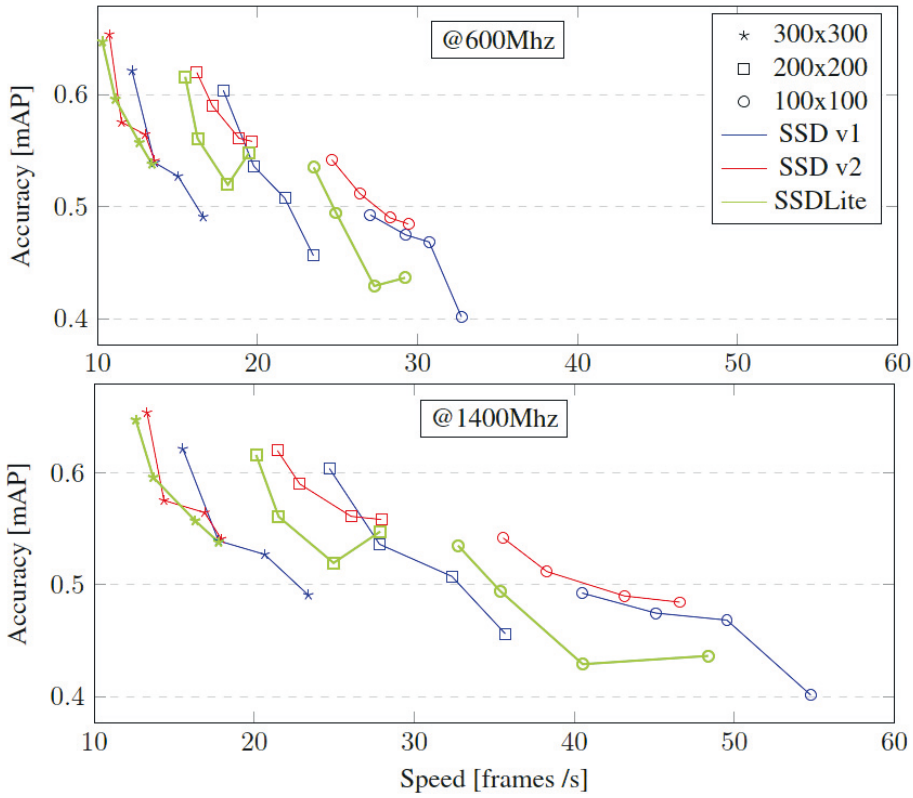


Figure 4.3 The effect of different depth multipliers, networks, input size, and CPU clock frequency is experimented with different Single Shot Multibox Detector (SSD) models.

possibility of performing simultaneous tasks on low-cost hardware, improving privacy. The privacy is preserved by saving only the features generated by the network, which are not enough for the reconstruction of the input image. With recent advancements in technology, small edge devices are getting more processing power. Consider will increase the value of the proposed approach to design systems for a broader range of real-world applications.

4.2 Real-time Facial Attribute Recognition System

Face detection is one of the most explored fields in computer vision, with many research works starting from the early 90s. Recent face recognition applications utilized deep learning object detectors as their fundamental building blocks. Researchers have been focusing primarily on the accuracy and speed of the single-task networks. In publication V, we focused on the practical approach of developing a multi-functional facial analysis system. The paper presents a real-time facial analysis demo running on a single desktop and evaluates many design options, such as what neural network models to use for each facial analysis task. We study to solve two research questions: (1) how to design a system with multiple tasks networks for a complete real-time facial attribute recognition, and (2) how to achieve smooth real-time performance with different design spectrums for the optimal design and combination of tasks.

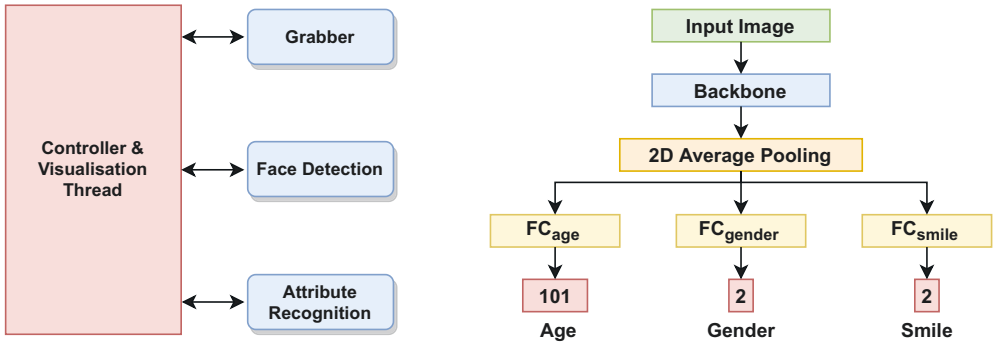
Multitask learning is getting popular and has been proven effective in many computer vision problems [107, 111, 114]. Ranjan *et al.* [70] proposed a unified deep multitask network for face detection, landmark localization, pose estimation, and gender recognition. In [71], authors further added smile, age, and face recognition tasks on multitask learning frameworks. Our multitask method presented in Figure 4.4 (right) is similar to this work. Our multitask network uses transfer learning to customize pretrained network weights for age, gender, and smile recognition tasks. Users can freely select any pretrained network weight to fine-tune the multitask network for robust performance.

Moreover, we present a complete architecture consisting of relevant tasks and efficiently design a unified system to achieve real-time performance on resource-limited devices. To our best knowledge, there have not been articles describing system-level functionality as a single system instead of focusing on each task.

In publication VI, we study the practical implementation of facial attributes recognition as a single system. Moreover, we develop a complete facial analysis system consisting of detection, alignment, age, gender and smile recognition and feature matching tasks using the available stage-of-the-art of-the-shelf networks for each task.

The contributions of this work are as follows:

- We presented a detailed system-level architecture for estimating several attributes from facial images and studied the design spectrum for real-time perfor-



(a) Software architecture

(b) Multitask classification network

Figure 4.4 Software architecture of the proposed facial recognition system in (a) and the structure of our multitask classification network in (b). The network is able to classify age, gender, and smile attributes for a given image by utilizing a common backbone network. Reproduced with the permission from publication VI

mance on the CPU.

- We discovered that it is better to use the multitask network than single-task networks. With the proposed multi-threaded architecture and multitask network (depicted in Figure 4.4), smooth inference speed is achieved on both CPU and GPU as illustrated in Table 4.1. Keep in mind the reported numbers are for running a single network at a time on the platform. So, running all three single-task network at the same time would result in a lower overall FPS for all of them due to limited computational power.
- For the reference to the community, all relevant information, source codes, and trained models are released publicly with detailed implementation instructions.

Table 4.2 shows the inference speed obtained by different combinations of face detection and attribute recognition networks. We experimented with a faster RCNN detector and a multitask network with MobileNet V1 backbone in the first row. The SSD detector with MobileNet v1 backbone is used with three MobileNet networks for age, gender, and smile recognition in the second row. On the third row same detection model was used with EfficientNet B0 multitask network. Finally, a combination of the fastest models, SSDlite and MobileNet V1 multitask network, is tested.

We present the design of a real-time facial analysis system. For a given real-time

Table 4.1 Accuracies and inference speed at different stages in our system with single output networks and multitask networks. The value of depth multiplier $\alpha = 1.0$ is used in all MobileNetV1 networks.

Stage Network	Accuracy	FPS(CPU)	FPS(1050TI)	FPS(1080TI)
Age MobileNetV1	4.9 MAE	31.61	148.90	147.44
Gender MobileNetV1	88.3%	31.48	150.45	149.75
Smile MobileNetV1	87.2%	31.46	148.84	148.78
Multitask MobileNetV1	5.67 MAE 84.2% Gender 83.6% Smile	29.80	147.06	147.20
Multitask EfficientNetB0	5.35 MAE 87.5% Gender 86.0% Smile	25.61	143.35	144.24

Table 4.2 Inference speed of our system with different combinations of networks on two environments. Average of three tests with video clips consisting of 1 – 12 faces in each frame.

Network combinations	FPS(CPU)	FPS(1050TI)
RCNN + multitask (MobileNet v1)	1.24	2.59
SSD + 3 x MobileNet v1	7.22	9.83
SSD + multitask (EfficientNet B0)	9.33	13.65
SSDlite + multitask (MobileNet v1)	**15.20**	**21.48**

video stream from a camera, the system performs the pipeline of face detection, alignment, gender and age estimation, smile recognition, and extracts neural network-based face representations and matches against a gallery database by nearest neighbor search. This system is an asynchronous design that consists of multiple threads/modules running on CPU and GPU. These modules are implemented using OpenCV and employ off-the-shelf neural network models for face detection, facial attribute recognition, and face verification, e.g., SSD, ResNet, MobileNet V1 and V2. The system also experiments with many well-known neural network models and evaluates their

performance on a celebrity search demo using the CelebA database with over 10K identities. The paper presents detailed design of a real-time facial analysis system (demo) running on a single desktop computer and evaluates many design options, such as what neural network models to use for each facial analysis task.

4.3 Trajectory Prediction for Mobile Objects Using Object Detection

The autonomous movement of a robot requires three main components: (1) a localization system that allows the robot to determine its location either in a local frame or a global frame, (2) a perception system that allows the robot to detect any obstacles in its current path, and (3) a trajectory planning algorithm that allows the robot to reach the destination with minimum cost while avoiding any collision

Designing a suitable perception system is challenging. The system needs a way of detecting any obstacles in its field of view. Some standard tools used in such systems are laser-based (such as LIDAR), sonar-based (such as Radar), or camera-based. Laser-based tools often suffer from the presence of sunlight due to their nature of using infrared lights. On the other hand, sonar-based tools cannot pinpoint objects due to the nature of wave propagation. Thus, camera-based methods have been getting more attention recently. Object detecting is an AI method to detect objects of interest in an image or a video. It can be used to find obstacles and track them in real-time to help the trajectory planning algorithm.

We deployed an object detector on an embedded device to collect the information from the real-world environment for trajectory prediction of moving objects. For the data collection, a Raspberry Pi is installed with a camera module to collect full HD videos over different times of the day during the summer months. Image frames are extracted from these videos and labeled for object detection training using the method presented in publication I. Both one-stage and two-stage detection networks are fine-tuned on this dataset using the pretrained network from the COCO dataset. We then deployed a one-stage object detector on a stationary Raspberry Pi equipped with a camera module. The detection system is then set up to detect moving objects in the test environment and collect the required data for trajectory prediction. As the system uses a stationary monocular camera, the distance to the detected object

is unknown. A linear transformation matrix is fitted on the actual coordinates of some objects to be utilized for the distance estimation as mentioned in Publication VII.

Moreover, we collected object positions relative to the camera fame and detection timestamp from the trajectory prediction systems. A linear transformation is fitted to map the camera points and the map coordinates by minimizing the euclidean distance between the known map points and the relative position in the image. The detected object is projected to the map by applying the transformation to the center points between the two bottom corners of their bounding boxes. The trajectory prediction library is then used to estimate the future path of the moving objects.

The trajectory prediction using a path library consists of several tasks. The vision-based trajectory prediction system consists of several tasks. The tasks and the structure design of their connections are represented in Figure 4.5.

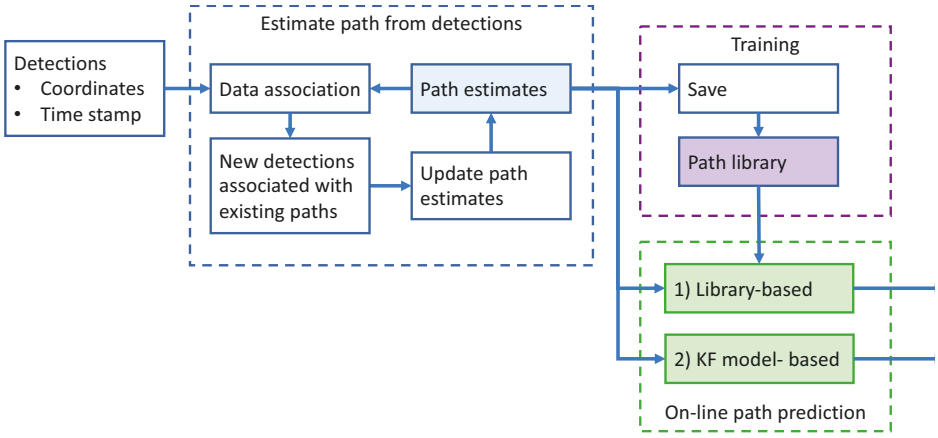


Figure 4.5 Building blocks of our trajectory prediction system. Reproduced with the permission from publication VII.

We study the operation of a situational awareness system where data is collected from a deep neural network-based object detector deployed on low-resourced camera-equipped Raspberry PI. The detection algorithm detects moving objects on the camera feed and stores only the coordinates and timestamps of the detected objects.

The contributions of this work are as follows:

- We present a privacy-preserving instance-based trajectory prediction for mobile objects based on the trajectory prediction algorithms.
- We use an inexpensive object detection network on a single board edge device

to collect moving objects' locations from the camera field of view.

We present a privacy-aware trajectory prediction method based on a path library learned from the mobile objects inside the surveillance area. The method starts by comparing the most recent path of the objects in the scene to each individual instance in the library. Each path gets a weight based on a similarity metric, and the future path is predicted by calculating a weighted average. Overall, our method uses probabilistic reasoning based on joint probabilistic data association, Hungarian algorithm, and Kalman filter to infer which detections from different time instances came from the same object.

5 CONCLUSIONS

Modern object detectors require a large number of labeled samples to tune millions of parameters correctly, are computationally heavy, and usually require powerful computing devices for training and inference. This dissertation considers two distinct aspects of training and deploying CNN-based object detectors: labor-intensive image annotation and efficient deployment for applications. Object labeling has been a well-known challenge for deep learning object detectors. In this thesis, we propose efficient methods for image data annotation for bounding box detection. The computer-assisted data annotation approaches effectively utilize human-machine collaboration to reduce the required time and effort to improve the object labeling task. The second part of the thesis includes efficient implementation methods for applying object detection networks on imaging pipelines for diverse applications. Following we provide the explicit answer to research questions:

Research Question 1: How sensitive existing object detectors are to label noise and how to improve the robustness of the detectors?

We studied the robustness of the one-stage object detectors on missing labels. We proposed a simple fix without much bell and whistle on detection network design. Our experiments showed that the object detector performance on noisy training data could be effectively increased with particular attention on selecting the loss function and the hyper-parameters.

Research Question 2: How to use existing machine learning and human annotator efficiently to label training data for supervised object detectors?

We proposed a two-stage scheme to create bounding box labels to train object detectors. Instead of drawing bounding boxes from scratch for all images, the annotator only needs to draw bounding boxes on some portion of the dataset while verifying and correcting the rest of the label proposals. Additionally, a fully labeled object detection dataset from indoor premises has been collected, labeled, and published for registration-free use.

Furthermore, we extended the above-mentioned two-stage scheme to an iterative loop for detector training and labeling. The object detector training, label proposals, inspection, and correction works are done in a close loop, taking a small batch of unlabeled images per cycle. This scheme has the benefit of providing space for sample selection over the small set of images; the inspection and correction work is relatively more straightforward as there are fewer images to handle. The detector is getting better in each iteration which typically proposes more accurate bounding boxes for unlabeled images and simplifies the human annotator work.

Finally, we extended the iterative train-annotate loop with sample selection. Previous works have shown that training object detectors with the more informative images to be labeled significantly impact the detection performance and the overall image labeling task. To this end, we studied the sample selection and proposed a similarity-based approach for efficient sample selection for bounding box annotations for object detector training.

Research Question 3: How to integrate object detection networks into an image-processing pipeline for the resource-limited platforms and put them into production at scale?

We studied the practical issues of deploying deep learning-based object detectors on computer vision applications. We deployed object detectors efficiently on the resource constraint devices while achieving smooth operation in multi-function applications. Multiple object detection networks were studied and deployed on less powerful single board edge devices for human tracking systems to predict future paths of moving objects in indoor and outdoor premises. Furthermore, we developed a real-time facial attribute recognition system in a multi-threaded multitask approach. It was tested on well-known neural network backbones. We evaluated their performance on face detection, age, gender, and smile recognition and similarity search based on the nearest neighbor using the CelebA database with over 10K identities.

Overall, we have discovered that one of the main issues with training CNN-based object detectors is the quality of annotated data and the existence of label noise. We proposed a method for iterative annotation of datasets to reduce the cost and mental pressure on the annotator. This in turn will result in a cleaner dataset as the annotator does not have to spend as much energy and money to get the data ready.

We also discovered the lack of efficient implementation approaches for object de-

tectors on low-resource edge devices. These devices are typically used as a means of rapid prototyping and production at scale, which means a proper implementation approach would result in a better quality of the end product. We designed a multi-task network suitable for such devices and illustrated its effectiveness over single task networks.

Future Work

The use of AI is rapidly increasing within sub-fields and inter-disciplinary fields alike. This thesis has discussed the two challenges of the deep learning object detectors, annotation of training datasets and efficient implementation on embedded systems. These challenges have been studied for some time now, and progress is being made with every new research outcome. In this part, we would like to present some ideas that might be worth investigating for future work.

The commercial and non-commercial tools and service providers for the data labeling are commonly available in the market. However, there is always some concern about the cost, data sharing, and privacy of data annotation with third-party services. While the can use the method proposed in this thesis to preserve the privacy of the user data, it does not incoherently provide that functionality. It may be worth exploring whether one can build an annotation tool that provides some level of privacy.

While publication IV tackled the idea of a noise-robust network, it only covered a specific type of noise and proposed a simple fix by tuning the hyperparameters. Due to the importance of robustness and its relation to safety, it is worth investigating other types of noise and finding a more sophisticated approach to counter their effect while preserving the performance.

REFERENCES

[1] D. Acuna, H. Ling, A. Kar and S. Fidler. Efficient Interactive Annotation of Segmentation Datasets With Polygon-RNN++. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 859–868.

[2] B. Adhikari and H. Huttunen. Iterative Bounding Box Annotation for Object Detection. *International Conference on Pattern Recognition (ICPR)*. 2021, 4040–4046. DOI: 10.1109/ICPR48806.2021.9412956.

[3] B. Adhikari, X. Ni, E. Rahtu and H. Huttunen. Towards a Real-time System for Facial Analysis. *IEEE International Workshop on Multimedia Signal Processing (MMSP)*. 2021. DOI: 10.1109/MMSP53017.2021.9733663.

[4] B. Adhikari, J. Peltomaki, J. Puura and H. Huttunen. Faster Bounding Box Annotation for Object Detection in Indoor Scenes. *European Workshop on Visual Information Processing (EUVIP)*. 2018. DOI: 10.1109/EUVIP.2018.8611732.

[5] B. Adhikari, J. Peltomäki, S. B. Germi, E. Rahtu and H. Huttunen. Effect of Label Noise on Robustness of Deep Neural Network Object Detectors. *Computer Safety, Reliability, and Security. SAFECOMP Workshops*. Ed. by I. Habli, M. Sujan, S. Gerasimou, E. Schoitsch and F. Bitsch. 2021, 239–250. DOI: 10.1007/978-3-030-83906-2_19.

[6] B. Adhikari, E. Rahtu and H. Huttunen. Sample Selection for Efficient Image Annotation. *European Workshop on Visual Information Processing (EUVIP)*. 2021. DOI: 10.1109/EUVIP50544.2021.9484022.

[7] N. Agarwal, A. Krohn-Grimberghe and R. Vyas. Facial Key Points Detection using Deep Convolutional Neural Network-NaimishNet. *arXiv preprint arXiv:1710.00977* (2017).

[8] B. Alexe, T. Deselaers and V. Ferrari. What is an object?: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, 73–80. DOI: 10.1109/CVPR.2010.5540226.

[9] A. Berg, J. Johnander, F. Durand de Gevigney, J. Ahlberg and M. Felsberg. Semi-Automatic Annotation of Objects in Visual-Thermal Video. *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2019.

[10] L. Beyer, X. Zhai, A. Oliver and A. Kolesnikov. S4L: Self-Supervised Semi-Supervised Learning. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2019, 1476–1485.

[11] S. Budd, E. C. Robinson and B. Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis* 71 (2021), 102062. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2021.102062. URL: https://www.sciencedirect.com/science/article/pii/S1361841521001080.

[12] L. Castrejón, K. Kundu, R. Urtasun and S. Fidler. Annotating Object Instances with a Polygon-RNN. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[13] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler and R. Urtasun. Monocular 3D Object Detection for Autonomous Driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[14] F. R. Cordeiro and G. Carneiro. A Survey on Deep Learning with Noisy Labels:How to train your model when you cannot trust on the annotations?: *Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2020.

[15] I. Croitoru, S.-V. Bogolin and M. Leordeanu. Unsupervised Learning from Video to Detect Foreground Objects in Single Images. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017, 4345–4353. DOI: 10.1109/ICCV.2017.465.

[16] Z. Dai, B. Cai, Y. Lin and J. Chen. UP-DETR: Unsupervised Pre-Training for Object Detection With Transformers. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, 1601–1610.

[17] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2005.

[18] S. V. Desai, A. L. Chandra, W. Guo, S. Ninomiya and V. N. Balasubramanian. An Adaptive Supervision Framework for Active Learning in Object Detection. *British Machine Vision Conference (BMVC)*. 2019.

[19] A. Dutta and A. Zisserman. The VIA Annotation Software for Images, Audio and Video. *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. New York, NY, USA: ACM, 2019. DOI: 10.1145/3343031.3350535.

[20] S. Escalera, M. T. Torres, B. Martínez, X. Baró, H. J. Escalante, I. Guyon, G. Tzimiropoulos, C. Corneanu, M. Oliu, M. A. Bagheri and M. Valstar. ChaLearn Looking at People and Faces of the World: Face Analysis Workshop and Challenge 2016. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2016.

[21] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision* (2008).

[22] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision* 111 (2015), 98–136.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010).

[24] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems* 25.5 (2014), 845–869.

[25] A. Geiger, P. Lenz and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[26] R. Girshick. Fast R-CNN. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, 1440–1448.

[27] R. Girshick, J. Donahue, T. Darrell and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, 580–587.

[28] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár and K. He. *Detectron*. `https://github.com/facebookresearch/detectron`. 2018.

[29] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. MIT Press, 2016.

[30] K. He, G. Gkioxari, P. Dollár and R. Girshick. Mask R-CNN. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017.

[31] K. He, X. Zhang, S. Ren and J. Sun. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[33] V. Jain and E. Learned-Miller. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. Tech. rep. University of Massachusetts, Amherst, 2010.

[34] J. Jeong, V. Verma, M. Hyun, J. Kannala and N. Kwak. Interpolation-Based Semi-Supervised Learning for Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, 11602–11611.

[35] L. Jiang, Z. Zhou, T. Leung, L.-J. Li and L. Fei-Fei. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, 2304–2313.

[36] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng and R. Qu. A Survey of Deep Learning-Based Object Detection. *IEEE Access* 7 (2019), 128837–128868. DOI: `10.1109/access.2019.2939201`.

[37] L. Jing and Y. Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[38] D. E. King. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research* 10 (2009), 1755–1758.

[39] T. Kiyokawa, K. Tomochika, J. Takamatsu and T. Ogasawara. Fully Automated Annotation with Noise Masked Visual Markers for Deep Learning Based Object Detection. *IEEE Robotics and Automation Letters* 4 (2019).

[40] K. Konyushkova, J. R. R. Uijlings, C. H. Lampert and V. Ferrari. Learning Intelligent Dialogs for Bounding Box Annotation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, 9175–9184.

[41] M. Kragh, R. N. Jørgensen and H. Pedersen. Object Detection and Terrain Classification in Agricultural Fields Using 3D Lidar Data. *Computer Vision Systems*. Springer, 2015, 188–197.

[42] A. Krizhevsky, I. Sutskever and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60 (2012), 84–90.

[43] F. C. Kurnaz, HocaõluBurak, M. K. Yılmaz, İ. Sülo and S. Kalkan. ALET (Automated Labeling of Equipment and Tools): A Dataset for Tool Detection and Human Worker Safety Detection. (2020), 371–386. DOI: 10.1007/978-3-030-66823-5_22.

[44] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982* (2018).

[45] K.-H. Lee, X. He, L. Zhang and L. Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 5447–5456.

[46] H. Leppäkoski, B. Adhikari, L. Raivio and R. Ritala. Prediction of Future Paths of Mobile Objects Using Path Library. *Open Engineering* 11.1 (2021), 1048–1058. DOI: 10.1515/eng-2021-0103.

[47] C. Li, C. Zhang, K. Ding, G. Li, J. Cheng and H. Lu. BundleNet: Learning with Noisy Label via Sample Correlations. *IEEE Access* 6 (2018), 2367–2377.

[48] D. Li, J. Yang, K. Kreis, A. Torralba and S. Fidler. Semantic Segmentation with Generative Models: Semi-Supervised Learning and Strong Out-ofDomain Generalization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[49] X. Li and Y. Guo. Adaptive Active Learning for Image Classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, 859–866.

[50] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár. Focal Loss for Dense Object Detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[51] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick. Microsoft coco: Common objects in context. *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, 740–755.

[52] H. Ling, J. Gao, A. Kar, W. Chen and S. Fidler. Fast Interactive Object Annotation With Curve-GCN. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[53] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu and M. Pietikäinen. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision* 128.2 (2019), 261–318. DOI: 10.1007/s11263-019-01247-4.

[54] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg. SSD:Single shot multibox detector. *European conference on computer vision*. Springer. 2016, 21–37.

[55] Z. Liu, P. Luo, X. Wang and X. Tang. Deep Learning Face Attributes in the Wild. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2015.

[56] B. Lutnick, B. Ginley, D. Govind, S. D. McGarry, P. S. LaViolette, R. Yacoub, S. Jain, J. E. Tomaszewski, K.-Y. Jen and P. Sarder. An integrated iterative annotation technique for easing neural network training in medical image analysis. *Nature Machine Intelligence* (2019), 112–119.

[57] K. Madono, T. Nakano, T. Kobayashi and T. Ogawa. Efficient Human-In-The-Loop Object Detection using Bi-Directional Deep SORT and Annotation-Free Segment Identification. *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2020, 1226–1233.

[58] P. M. R. Martin Koestinger Paul Wohlhart and H. Bischof. Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*. 2011.

[59] J. Matas, O. Chum, M. Urban and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing* 22.10 (2004), 761–767. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2004.02.006.

[60] G. Menghani. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. abs/2106.08962 (2021). arXiv: 2106.08962.

[61] S. D. MPLab Uiversity of California. *The MPLab GENKI Database, GENKI-4K Subset*.

[62] OpenCV. *Open Source Computer Vision Library*. 2015. URL: https://opencv.org/.

[63] D. P. Papadopoulos, A. D. Clarke, F. Keller and V. Ferrari. Training object class detectors from eye tracking data. *European conference on computer vision*. Springer. 2014, 361–376.

[64] D. P. Papadopoulos, J. R. Uijlings, F. Keller and V. Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 854–863.

[65] D. P. Papadopoulos, J. R. Uijlings, F. Keller and V. Ferrari. Extreme clicking for efficient object annotation. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2017, 4940–4949.

[66] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller and V. Ferrari. Training Object Class Detectors with Click Supervision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 180–189.

[67] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer. Automatic differentiation in PyTorch. *NIPS-W*. 2017.

[68] Q. Qian, L. Chen, H. Li and R. Jin. DR Loss: Improving Object Detection by Distributional Ranking. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[69] E. Rahtu, J. Kannala and M. Blaschko. Learning a category independent object detection cascade. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011, 1052–1059. DOI: 10.1109/ICCV.2011.6126351.

[70] R. Ranjan, V. M. Patel and R. Chellappa. HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[71] R. Ranjan, S. Sankaranarayanan, C. D. Castillo and R. Chellappa. An All-In-One Convolutional Neural Network for Face Analysis. *12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*. 2017.

[72] J. Redmon, S. Divvala, R. Girshick and A. Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, 779–788.

[73] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, 6517–6525.

[74] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv* (2018).

[75] S. Ren, K. He, R. Girshick and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*. 2015, 91–99.

[76] F. Rothganger, S. Lazebnik, C. Schmid and J. Ponce. 3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2003, II–272. DOI: 10.1109/CVPR.2003.1211480.

[77] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115.3 (2015), 211–252.

[78] O. Russakovsky, L.-J. Li and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, 2121–2131.

[79] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, 4510–4520.

[80] F. Schroff, D. Kalenichenko and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[81] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

[82] A. Smailagic, P. Costa, A. Gaudio, K. Khandelwal, M. Mirshekari, J. Fagert, D. Walawalkar, S. Xu, A. Galdran, P. Zhang, A. Campilho and H. Noh. O-MedAL: Online active deep learning for medical image analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10 (2020).

[83] H. Su, J. Deng and L. Fei-fei. Crowdsourcing Annotations for Visual Object Detection. *AAAI Human Computation Workshop*. 2012, 40–46. DOI: 10.1145/2660114.2660119.

[84] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[85] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[86] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu. A Survey on Deep Transfer Learning. *Artificial Neural Networks and Machine Learning – ICANN 2018*. Ed. by V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis and I. Maglogiannis. Cham: Springer International Publishing, 2018, 270–279. ISBN: 978-3-030-01424-7.

[87] M. Tan and Q. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Proceedings of the 36th International Conference on Machine Learning (ICML)*. PMLR, 2019.

[88] M. Tan, R. Pang and Q. V. Le. EfficientDet: Scalable and Efficient Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[89] TensorFlow. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: `https://www.tensorflow.org/`.

[90] Z. Tian, C. Shen, H. Chen and T. He. FCOS: Fully Convolutional One-Stage Object Detection. *Proceedings of the International Conference on Computer Vision (ICCV)*. 2019.

[91] J. Tremblay, T. To and S. Birchfield. Falling Things: A Synthetic Dataset for 3D Object Detection and Pose Estimation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018.

[92] T. Tuytelaars and L. Van Gool. Matching Widely Separated Views Based on Affine Invariant Regions. *International Journal of Computer Vision* 59 (2004), 61–85. DOI: `10.1023/B:VISI.0000020671.28016.e8`.

[93] J. Uijlings, K. Sande, T. Gevers and A. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision* 104 (2013), 154–171. DOI: `10.1007/s11263-013-0620-5`.

[94] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2001.

[95] C.-Y. Wang, I.-H. Yeh and H.-Y. M. Liao. You Only Learn One Representation: Unified Network for Multiple Tasks. *arXiv preprint arXiv:2105.04206* (2021).

[96] O. Willers, S. Sudholt, S. Raafatnia and S. Abrecht. Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks. *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*. Cham: Springer International Publishing, 2020.

[97] V. W. H. Wong, M. Ferguson, K. H. Law and Y. T. Lee. An Assistive Learning Workflow on Annotating Images for Object Detection. *2019 IEEE International Conference on Big Data (Big Data)*. 2019, 1962–1970.

[98] E. Wozniak, C. Cârlan, E. Acar-Celik and H. J. Putzer. A Safety Case Pattern for Systems with Machine Learning Components. *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops*. Cham: Springer International Publishing, 2020.

[99] B. Wu, F. Iandola, P. H. Jin and K. Keutzer. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2017.

[100] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma and L. He. *A Survey of Human-in-the-loop for Machine Learning*. 2021. arXiv: 2108.00941.

[101] Z. Wu, N. Bodla, B. Singh, M. Najibi, R. Chellappa and L. Davis. Soft Sampling for Robust Object Detection. *British Machine Vision Conference (BMVC)*. 2019.

[102] Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh and B. Chen. MobileDets: Searching for Object Detection Architectures for Mobile Accelerators. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, 3825–3834.

[103] B. Yang, W. Luo and R. Urtasun. PIXOR: Real-Time 3D Object Detection From Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[104] A. Yao, J. Gall, C. Leistner and L. Van Gool. Interactive object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, 3242–3249. DOI: 10.1109/CVPR.2012.6248060.

[105] D. Yoo and I. S. Kweon. Learning Loss for Active Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[106] J. Yrjänäinen, X. Ni, B. Adhikari and H. Huttunen. Privacy-Aware Edge Computing System For People Tracking. *IEEE International Conference on*

Image Processing (ICIP). 2020, 2096–2100. DOI: 10.1109/ICIP40778.2020.9191260.

[107] X. Yuan, X. Liu and S. Yan. Visual Classification With Multitask Joint Sparse Representation. *IEEE Transactions on Image Processing* 21.10 (2012).

[108] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar and B. Lee. A Survey of Modern Deep Learning based Object Detection Models. *arXiv preprint arXiv:2104.11892* (2021).

[109] D. Zhang, S. Mishra, E. Brynjolfsson, J. Etchemendy, B. G. Deep Ganguli, T. Lyons, J. Manyika, J. C. Niebles, M. Sellitto, Y. Shoham, J. Clark and R. Perrault. The AI Index 2021 Annual Report. (2021).

[110] H. Zhang, F. Chen, Z. Shen, Q. Hao, C. Zhu and M. Savvides. Solving Missing-Annotation Object Detection with Background Recalibration Loss. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, 1888–1892.

[111] K. Zhang, Z. Zhang, Z. Li and Y. Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23 (2016).

[112] X.-Y. Zhang, C.-L. Liu and C. Y. Suen. Towards Robust Pattern Recognition: A Review. *Proceedings of the IEEE* 108.6 (2020), 894–922. DOI: 10.1109/JPROC.2020.2989782.

[113] Y. Zhang, H. Ling, J. Gao, K. Yin, J.-F. Lafleche, A. Barriuso, A. Torralba and S. Fidler. DatasetGAN: Efficient Labeled Data Factory With Minimal Human Effort. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, 10145–10155.

[114] Z. Zhang, P. Luo, C. C. Loy and X. Tang. Facial Landmark Detection by Deep Multi-task Learning. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.

[115] Q. Zhou, C. Yu, Z. Wang, Q. Qian and H. Li. Instant-Teaching: An End-to-End Semi-Supervised Object Detection Framework. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, 4081–4090.

[116] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review* 5.1 (2017), 44–53.

[117] C. Zitnick and P. Dollar. Edge Boxes : Locating Object Proposals from Edges. Vol. 8693. 2014. DOI: 10.1007/978-3-319-10602-1_26.

[118] Z. Zou, Z. Shi, Y. Guo and J. Ye. Object Detection in 20 Years: A Survey. *arXiv preprint arXiv:1905.05055* (2019).

PUBLICATIONS

PUBLICATION

I

Faster Bounding Box Annotation for Object Detection in Indoor Scenes

B. Adhikari, J. Peltomaki, J. Puura and H. Huttunen

European Workshop on Visual Information Processing (EUVIP), 2018
DOI: 10.1109/EUVIP.2018.8611732

Faster Bounding Box Annotation for Object Detection in Indoor Scenes

*Bishwo Adhikari, *Jukka Peltomäki, **Jussi Puura and *Heikki Huttunen
*Tampere University of Technology, Tampere, Finland
**Sandvik Mining and Construction Oyj, Tampere, Finland

***Abstract*—This paper proposes an approach for rapid bounding box annotation for object detection datasets. The procedure consists of two stages: The first step is to annotate a part of the dataset manually, and the second step proposes annotations for the remaining samples using a model trained with the first stage annotations. We experimentally study which first/second stage split minimizes to total workload. In addition, we introduce a new fully labeled object detection dataset collected from indoor scenes. Compared to other indoor datasets, our collection has more class categories, different backgrounds, lighting conditions, occlusion and high intra-class differences. We train deep learning based object detectors with a number of state-of-the-art models and compare them in terms of speed and accuracy. The fully annotated dataset is released freely available for the research community.**

***Index Terms*—Bounding box annotation, object detection, deep learning, indoor dataset**

I. Introduction

Object detection from images is a well-known area of research in machine learning and computer vision. Today, object detection algorithms have matured enough to solve real-world problems. Object detection is a central component in face detection, object counting, visual search, landmark recognition, satellite image analysis, autonomous driving, drone and agriculture production assessment [1–3]. Object detection is known to be a challenging task in computer vision as a large number of labeled datasets is needed for learning and generalization performance of the detection model.

The collection of a large scale dataset representative enough is a challenge and the key question in order to train detectors robust to variations in object appearance. Moreover, the annotation of large collections of images is both labor-intensive and error-prone. Traditionally, the annotation problem is solved by brute force, *i.e.,* by crowdsourcing a large group of annotators on a web platform such as the Amazon Mechanical Turk. Examples of some popular large-scale datasets for object detection with labeled data are ImageNet [4], MS COCO [5], and PASCAL VOC [6]. There are also public datasets for specific domains, such as face detection [7], character recognition [8], landmark recognition and detection [9], MCIndoor20000 [10] and Freiburg grocery dataset [11].

As machine learning methods and platforms develop into a more mature state, the focus is turning towards applications. In a small scale application project, the collection of data can not be scaled up by using thousands of annotators. Instead, there is a need for agile and rapid annotation procedures that enable the deployment of machine learning and object detection in small projects, as well. Like most machine learning topics, the most time-consuming task of object detection is also the annotation of each object area in the image dataset. For example, annotating the bounding boxes of a single image from the 14 million sample Imagenet [4] dataset takes 42 seconds per bounding-box by crowdsourcing using the Mechanical Turk annotation tools [12].

Fig. 1. Examples of object instances from TUT indoor dataset.

There have been many studies focusing on how to speed up the image dataset annotation such as box verification series [13], eye-tracking [14] and learning intelligent dialogs [15]. Moreover, semantic annotation of objects in image datasets has been widely discussed in recent years: Polygon RNN++ [16] and Extreme clicking [17] are recent proposals for efficient object segmentation on image datasets. However, bounding box annotation is by far the most common task in practical and industrial applications, and despite the aforementioned sophisticated approaches, their use is still not very common as easy-to-adopt robust used-friendly tools are missing.

In this work, we study a simple and practical heuristic to annotate the object bounding boxes of an image dataset. Our two-stage approach splits the data into two folds; the first fold is manually annotated from scratch, after which an object detector is trained for generating proposal annotations for the second fold. Thus, the total workload consists of the first stage annotations plus the corrections required to the

Fig. 2. Example of object instances from single class label (*Chair*). Intra-class difference between the object instances are high in *TUT indoor dataset*.

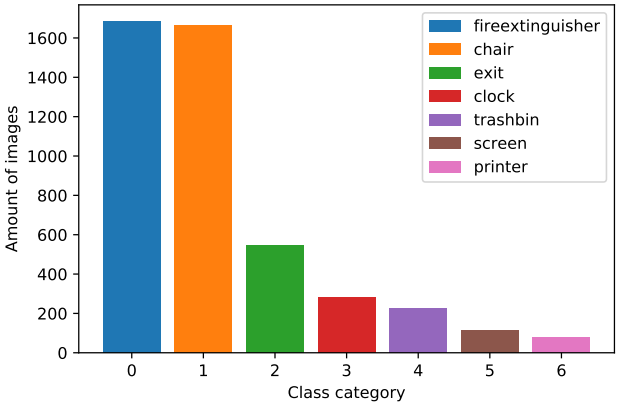


Fig. 3. Distribution of the class categories in the *TUT indoor dataset*. There are altogether 4595 instances from 7 categories.

second fold semi-automatically annotated proposals, and a natural question is to find the optimal split between the folds in order to minimize manual work.

The proposed method significantly reduces the workload to create big enough environment specific object detection dataset. In addition to the technique to fasten the bounding box annotation, we present fully annotated multiclass object detection dataset from indoor scenes, for which we have applied the proposed annotation procedure. Compared to other indoor datasets, our collection has more class categories, different backgrounds, lighting conditions, occlusion and high intra-class differences. Examples of objects in the dataset are shown in Figure 1.

The remainder of the paper is structured as follow: Section II will describe the dataset. Section III will describe the methods used to fasten the bounding box annotation on image dataset. Section IV will show the experiments and results obtained with them and comments on findings, and Section V will present the conclusion and future work.

II. Dataset

We have collected an indoor scene dataset to experiment the feasibility of our method. The dataset is created from sequences of videos that were recorded from different indoor premises of Tampere University of Technology (TUT). The *TUT indoor dataset* is a fully-labeled image dataset to facilitate the board use of image recognition and object detection in indoor scenarios. In addition to the labeled images, we provide the sequence of recorded videos from multiple places inside the university. Our recorded HD videos together with the fully annotated dataset are freely available here [1].

The TUT indoor dataset consists of 2213 image frames containing seven classes. In contrast to existing indoor datasets, our dataset includes a variety of background, lighting conditions, occlusion and high inter-class differences. As shown in Figure 2, the interclass variation is high inside each class category. The variability in terms of background, illumination, blurring, occlusion, pose and scale are main features of TUT indoor dataset.

[1]https://sites.google.com/view/bishwoadhikari/dataset

The size of each frame extracted from the HD video sequences is 1280×720 captured using a high-quality camera with optical image stabilization. There are altogether 2213 frames having 4595 object instances from 7 class categories. Each frame consists of several object instances from several class categories. The distribution of object categories is shown in Figure 3. The highest number of object instances are from *Fire extinguisher* class followed by *Chair* class having more than 1600 instances. While the least populated class category is *Printer* containing less than 100 object instances. The maximum number of object instances from a single class is 1684 and the minimum is 81 instances.

The number of image frames in this dataset can be increased by a factor of 3-4 by using common image data augmentation techniques such as adding noise, blurring, flipping and rotation. As the quality of object detection model and proposal bounding box annotation are directly related to the trained model and images to be annotated, one can improve the performance by improving the quality of bounding box annotation at first place and increasing the quantity of labeled dataset to train the object detection models.

III. Proposed Method for Semi-Automatic Annotation

In this section, we describe our method for the semi-automatic annotation workflow. The motivation behind the workflow is to minimize the total amount of manual work. The basic idea is to train a model on a small subset of the data, and use that model to predict annotations in the larger set, thus allowing the human annotator to only correct the incorrect predictions in most of the images. More specifically, the workflow consists of six steps as illustrated in Figure 4.

A. Annotation Procedure

First, the unlabeled dataset is split into two parts. The first (usually smaller) part is first manually annotated by a human from scratch. Then the first part is used to train an object detector, which is then used to predict annotations for the remaining part of the data. The predicted data is manually

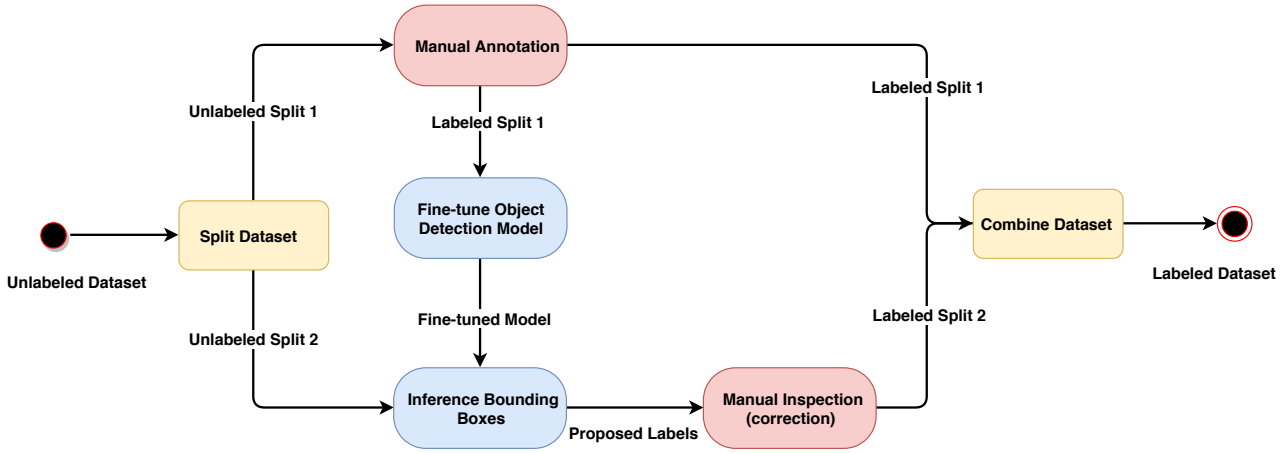


Fig. 4. The workflow of our semi-automatic bounding box image annotation method. The dataset is split into two parts. The first part is manually annotated and used to train a model, which is then used to predict labels on the rest of the dataset. After manually correcting the predicted labels, the final fully annotated dataset is combined from the manually annotated and corrected subsets.

corrected by a human, after which both parts of the split data are combined to form a fully labeled dataset. Note that two actions out of the outlined six are performed manually by a human, while the other actions can be fully automated. Although the approach is probably invented several times and in practical use already, a systematic evaluation of how this should be done has been dismissed until now. More specifically, we are interested in the proportions the two folds should have in order to minimize manual work and annotation time.

***Splitting the dataset*—**The first step in the workflow is to split the dataset to the train and test subsets. The split is performed within the individual video sequence of the TUT indoor dataset, so train and test subsets contain a similar ratio of images from each video sequence. The splitting within the sequences was done in the order of video progression, not randomly, so for instance the first image in every sequence was included in the train set of all the different folds. We experimented with train set folds of 1 % to 10 % with 1 % increases, and 15 % to 95 % with 5 % increases. The corresponding test set for each train set was always the remaining percentage, so a total of 100 % of the data was used in the experiment. Note that only a subset of the results are shown in the tables.

***Manual annotation of the train set*—**The second step is to fully annotate the unlabeled fold 1 dataset. This is done by a human. We used a basic bounding box annotation method with no extra speed up procedures.

***Training the detector*—**The third step is to train the detector. Although any detector can be used, we focus on the recent deep learning based object detection models, which we fine-tune using the manually annotated dataset. More specifically, we choose the Faster RCNN model using ResNet-101 network trained on the MS COCO dataset as our starting point. The pretrained models for object detection can be found on the TensorFlow Object Detection model zoo [2].

[2]https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

***Predicting annotations for the rest of the data*—**After fine-tuning the object detection model, it is used to predict the bounding boxes for the rest of the unlabeled dataset, the unlabeled fold 2.

***Manually correcting the inferred proposals*—**The inferred predictions are manually corrected by a human. The human needs to go through all the proposed labels, see if they make sense, and correct if necessary. Wrongly drawn boxes are removed, wrongly labeled classes are corrected and new boxes are drawn, if needed. We used the same annotation tool as when fully annotating.

***Combining the final labeled dataset*—**The fully annotated train fold and the fully corrected test fold are finally combined as the labeled dataset.

B. Estimating the Workload

The total amount of manual work consists of three kinds of manual operations:

1) Annotation of bounding boxes in the first fold,
2) Removal of false detection (false positives) in the second fold,
3) Addition of missed detections (false negatives) in the second fold.

We choose to define the false negatives and false positives by the amount of overlap between the true and predicted bounding box: We assume the user would correct the annotation if the *intersection-over-union* (IoU) overlap between the true object location and the predicted bounding box is less than 50 %.

Additionally, in the case of partial overlap less than 50 %, we model the user operation as removal of the incorrect box and addition of the box at the correct location. Alternatively, one might consider an additional user action: moving and/or resizing the box. However, this requires often more work (and mental attention) than simple removal and addition. Moreover, many annotation tools have not even implemented the box adjustment operation.

The workload required for the second fold corrections can be estimated from the precision and recall values of the first stage object detector. Precision is defined as

$$\text{precision} = \frac{\text{\# of correct detections}}{\text{\# of all detections}},$$

and recall as

$$\text{recall} = \frac{\text{\# of correct detections}}{\text{\# of all objects}}.$$

Using these two metrics, we can estimate what would be the workload for the tasks 1-3 above, respectively.

***First stage annotations*—**In the first stage, the user simply marks each object in each first fold image. Therefore, the number of manual operations is simply

$$\text{\# initial annotations} = \text{\# of true objects in fold 1}.$$

***Second stage additions*—**The recall value summarizes the proportion of true objects in the second stage fold that are correctly found by the detector. The complement of recall is then the proportion of objects *not* found by the detector. Thus, the number of additions the user would have to perform is given as

$$\text{\# additions} = (\text{\# of true objects}) \times (1 - \text{recall}).$$

***Second stage removals*—**On the other hand, the user has to remove all false detections, *i.e.,* detections at places with no true object. The precision value describes the proportion of detections that are in fact true objects. The complement of precision is then the proportion of detections that do *not* correspond to a true object, and needs to be manually removed. Thus, the number of removals the user would have to perform is given as

$$\text{\# removals} = (\text{\# of all detections}) \times (1 - \text{precision}).$$

The total work is the sum of these three steps:

$$\text{operations} = \text{initial annotations} + \text{additions} + \text{removals}.$$

However, we will see that the amount of time required for the different tasks varies a lot. In particular, the box removal is very fast, while the annotations from scratch take more time. Therefore, we also define the total working time required:

$$\text{time} = t_1\,(\text{initial annotations}) + t_2\,(\text{additions} + \text{removals})\,,$$

where t_1 denotes the time required for a single 1st stage annotation, and t_2 is the average time for a single 2nd stage correction.

IV. Experimental Results

Our interest is to estimate the workload needed to create a fully annotated environment specific multiclass object detection dataset. However, as we are also introducing a new dataset not previously studied, we will first assess its difficulty by training state-of-the-art object detectors and evaluate their accuracy. Understanding the accuracies of different models will also help to evaluate the impact of choosing a poor object detector for the second stage proposal generation. After that, we will study the annotation workload that would be required to annotate the data using different strategies.[3]

TABLE I
PERFORMANCE EVALUATION OF FINE-TUNED TENSORFLOW OBJECT DETECTION API MODELS AND RETINANET MODELS USING TUT INDOOR DATASET.

Model	mAP %	Speed (FPS)
TensorFlow SSD MobileNet	**97.73**	**27.41**
TensorFlow Faster RCNN Resnet50	96.54	11.80
TensorFlow Faster RCNN Resnet101	95.06	9.69
RetinaNet, backbone = Resnet50	96.60	13.00
RetinaNet, backbone = Resnet101	95.93	9.64
RetinaNet, backbone = VGG16	95.67	9.79
RetinaNet, backbone = VGG19	94.57	8.66

TABLE II
THE TOTAL MANUAL ANNOTATION WORKLOAD AT DIFFERENT DATASET SPLITTING RATIOS. A BIGGER TRAINING SET RESULTS IN BETTER INFERENCE PERFORMANCE, AS INDICATED BY THE MAP. FOR OUR DATASET, THE BEST WORKLOAD IS FOUND WHEN THE TRAINING SET SIZE IS BETWEEN ABOUT 4% - 10%.

Split	Workload	COCO mAP % @ IoU 0.5
Train 1 %, Test 99 %	1642	56.88
Train 2 %, Test 98 %	1118	67.45
Train 3 %, Test 97 %	970	75.99
Train 4 %, Test 96 %	915	77.75
Train 5 %, Test 95 %	943	80.18
Train 6 %, Test 94 %	863	81.63
Train 7 %, Test 93 %	938	82.56
Train 8 %, Test 92 %	979	85.46
Train 9 %, Test 91 %	931	79.96
Train 10 %, Test 90 %	963	82.72
Train 20 %, Test 80 %	1265	88.80
Train 40 %, Test 60 %	2016	95.42
Train 60 %, Test 40 %	2878	96.76
Train 80 %, Test 20 %	3704	96.65

***Accuracy on the TUT indoor dataset*—**For this study, we use three recent object detection pipelines. First one is the Regions-CNN (R-CNN) framework proposed by Ren *et al.* in 2015 [18], which uses a two-stage structure: first stage network creates object proposals, which the second stage network then classifies into different categories. The second approach is the Single Shot Detector (SSD) framework proposed in 2016 by Liu *et al.* [19]. Instead of the two-stage structure, SSD uses only single feedforward network to predict object locations (regression) and categories (classification) is a single network. Finally, we consider the *RetinaNet* structure proposed by Lin *et al.* in 2017 [20]. The RetinaNet extends the SSD approach by defining a novel loss function in order to focus the attention to the most difficult cases instead of the easy ones.

For all cases, we start training from MS COCO trained network and fine-tune using our own data. For the R-CNN and SSD structures we use the TensorFlow object detec-

[3]To clarify: the dataset is fully annotated, but we will compare different strategies how this could have been done faster.

tion framework [21] and for RetinaNet we use the Keras-RetinaNet library[4].

The accuracies of the experimented models are shown in Table I. In this experiment, 80 % of the total dataset is used for training the model and the rest is used for evaluating the performance of the detector. The mean average precision (mAP) calculation in our experiment follows the MS COCO [5] evaluation procedure, *i.e.,* we use a fixed set of 101 detection thresholds $t = 0, 0.01, \ldots, 1.0$ with no interpolation and average the precision metrics over all thresholds. We consider matches only for the IoU value of 0.5 and over, and not averaged over several IoUs as in the typical COCO evaluation metric.

According to the results of Table I, the simplest SSD network seems to have both the highest accuracy and highest computational speed. This is somewhat surprising as two-stage detectors (R-CNN) tend to have higher accuracy than the single-stage detectors [22]. On the other hand, this is good news since there is no need to consider the tradeoff between the speed and accuracy.

Workload minimization—The workload calculation based on different train-test splits is summarized in Table II. In this case, we use the Faster RCNN with ResNet101 for generating the second stage proposals. We decide to use this network instead of the fastest and more accurate SSD network because the SSD performance might be slightly anomalous and specific to our dataset only. Moreover, the performance differences are minor, so essentially any of structure would produce more or less similar results. In each case, we train altogether 50000 epochs.

The total manual workload to create a fully annotated dataset is the sum of manual workload for annotating the first dataset split and the workload needed for the correction. To minimize the workload, the initial annotation workload should be as low as possible, and the quality of the model as high as possible. The optimal minimized workload is found when these two conflicting requirements are both satisfied well enough.

To make the workload figures more intuitive, we also consider the time spent in the manual stages of the annotation process, which are based on the workload figures. From our timing experiment, we found that manual annotation takes about 10.15 seconds per bounding box and correction (addition and removal) takes about 5.20 seconds per bounding box. (In the timing experiment we used 80% of total dataset on training the model and remaining 20% to predict the bounding boxes). We are using these approximate times to estimate the time needed to draw all together 4595 bounding boxes in all image frames in TUT indoor dataset.

The calculation of the total time for the annotation is:

$$\text{Total time for annotation} = W_M \times 10.15 + W_C \times 5.20$$

Where, W_M is number of boxes required to annotate in the first fold of the dataset and W_C is workload for the correction of proposed bounding boxes in fold 2.

[4]https://github.com/fizyr/keras-retinanet

Figure 5 summarizes the results of work required to annotate the whole dataset. On the left, we show the number of bounding boxes to annotate in the two stages, as a function of the proportion of first/second stage folds. One can see that with very small first stage fold, the first stage becomes very easy to annotate, but the second stage proposals become unreliable and more work is needed on the second stage. On the other hand, if excessive amounts of training data is allocated to the first stage, then the total workload is dominated by the first stage, while the seconds stage requires virtually no corrections. The sweet spot is at a relatively low percentage: The best strategy is to annotate only 4–8% from scratch and use the trained model for the rest.

In the right panel of Figure 5 we show the time used for the total annotation in different split proportions, *i.e.,* we multiply the number of boxes with their estimated execution times. This has the effect of moving the sweet spot even further to the left, with the minimum at 4%.

In all splits, our method decreases the manual workload to annotate the full dataset significantly. Based on our measured annotation time, the total time needed to annotate the full TUT indoor dataset manually from scratch would be $4595 \times 10.15 = 46639.25$ seconds $\approx$ 13 hours. Using the proposed two-stage approach the required time decreases to less than two hours of manual labor.Also, the total work time starts to increase quite linearly with train set splits higher than 10%. Note that the time estimations include only the manual labor done by humans, and do not include the automated computing time between the manual phases of the task.

V. Conclusions

In this paper, we introduced the efficient way to annotate the object detection dataset using fine-tuned object detection model. We introduced a fully labeled TUT indoor dataset for object detection. Our dataset have 2213 image frames extracted from 6 different sequences of recorded videos containing 7 classes of common indoor objects. We tested the relevance of the dataset with current state-of-the-art object detection frameworks. The real-world indoor dataset is valuable resource for indoor object detection and handy for the fast experiment on object detection algorithms.

The main contribution of the paper is a two-stage procedure for rapid annotation of bounding boxes, together with a systematic assessment of the time savings introduced by the approach. It was found that the proposed two-stage method reduces manual work by almost 90 % and is cost efficient for the implementation of supervised object annotation campaigns.

We experimented that manual annotation takes about 10.15 seconds per bounding box while the correction takes about 5.20 seconds per box. Note that this calculation might differ in different scenarios such as different object types, changes in number of classes and annotation tools. It is found that the workload to annotate TUT indoor dataset is minimum when using 4–8% of the total dataset to fine-tuned the object detection model.

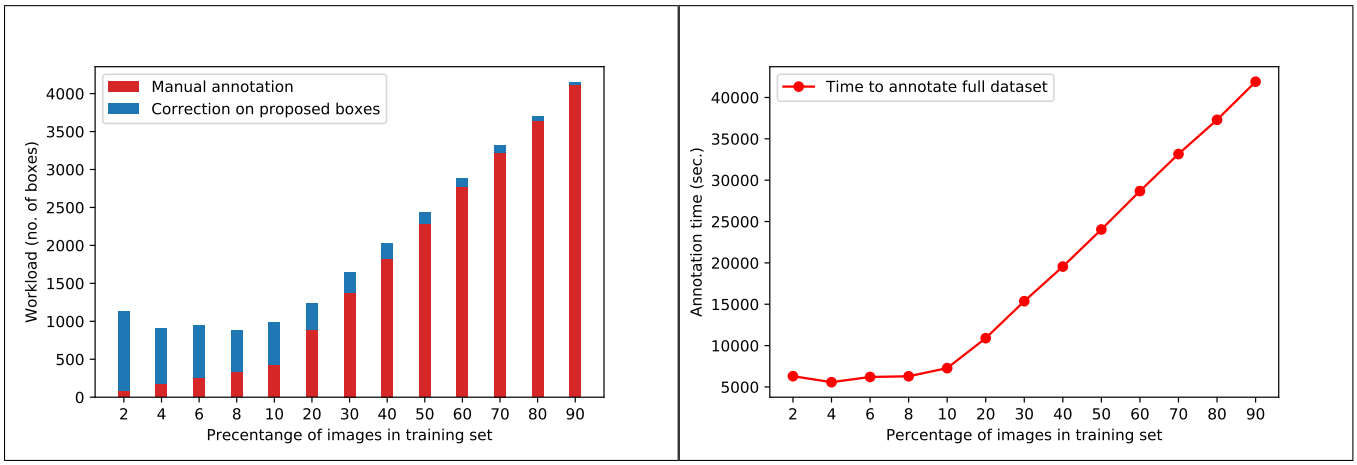


Fig. 5. Amount of workload needed in different train-test split (left). The proportion of annotation time needed to annotate full dataset using different portion of manually annotated data to fine-tuned the object detection model (right).

For future work, we plan to experiment more with other datasets to see how the procedure generalizes to larger number of classes, for instance. Secondly, instead of two stages, we plan to generalize the approach to more stages, incrementally improving the proposal model accuracy at each stage.

ACKNOWLEDGMENT

The authors would also like to thank CSC - The IT Center for Science for the use of their computational resources. The work was partially funded by the Academy of Finland project *309903 CoefNet* and Business Finland project *408/31/2018 MIDAS*.

REFERENCES

[1] M. Kragh, R. N. Jørgensen, and H. Pedersen, "Object detection and terrain classification in agricultural fields using 3d lidar data," in *Computer Vision Systems*, L. Nalpantidis, V. Krüger, J.-O. Eklundh, and A. Gasteratos, Eds. Cham: Springer International Publishing, 2015, pp. 188–197.

[2] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *CoRR*, vol. abs/1603.06201, 2016. [Online]. Available: http://arxiv.org/abs/1603.06201

[3] A. Teichman and S. Thrun, "Practical object recognition in autonomous driving and beyond," in *Advanced Robotics and its Social Impacts*, Oct 2011, pp. 35–38.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[5] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *Computing Research Repository*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[6] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan 2015. [Online]. Available: https://doi.org/10.1007/s11263-014-0733-5

[7] V. Jain and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," University of Massachusetts, Amherst, Tech. Rep. UM-CS-2010-009, 2010.

[8] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[9] H. Noh, A. Araujo, J. Sim, and B. Han, "Image retrieval with deep local features and attention-based keypoints," *CoRR*, vol. abs/1612.06321, 2016. [Online]. Available: http://arxiv.org/abs/1612.06321

[10] F. S.Bashiriab, E. LaRoseb, P. Peissigb, and A. P.Taftib, "Mcindoor20000: A fully-labeled image dataset to advance indoor objects detection," 2018. [Online]. Available: https://doi.org/10.1016/j.dib.2017.12.047

[11] P. Jund, N. Abdo, A. Eitel, and W. Burgard, "The freiburg groceries dataset," *CoRR*, vol. abs/1611.05799, 2016. [Online]. Available: http://arxiv.org/abs/1611.05799

[12] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing annotations for visual object detection," 2012. [Online]. Available: https://www.aaai.org/ocs/index.php/WS/AAAIW12/paper/view/5350

[13] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," *CoRR*, vol. abs/1602.08405, 2016. [Online]. Available: http://arxiv.org/abs/1602.08405

[14] D. P. Papadopoulos, A. D. F. Clarke, F. Keller, and V. Ferrari, "Training object class detectors from eye tracking data," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 361–376.

[15] K. Konyushkova, J. R. R. Uijlings, C. H. Lampert, and V. Ferrari, "Learning intelligent dialogs for bounding box annotation," *CoRR*, vol. abs/1712.08087, 2017. [Online]. Available: http://arxiv.org/abs/1712.08087

[16] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-rnn++," *CoRR*, vol. abs/1803.09693, 2018. [Online]. Available: http://arxiv.org/abs/1803.09693

[17] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari, "Extreme clicking for efficient object annotation," *CoRR*, vol. abs/1708.02750, 2017. [Online]. Available: http://arxiv.org/abs/1708.02750

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE International Conf. on Computer Vision, ICCV*, 2017.

[21] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *CoRR*, vol. abs/1611.10012, 2016. [Online]. Available: http://arxiv.org/abs/1611.10012

[22] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

PUBLICATION

II

Iterative Bounding Box Annotation for Object Detection

B. Adhikari and H. Huttunen

International Conference on Pattern Recognition (ICPR), 2021, 4040–4046
DOI: 10.1109/ICPR48806.2021.9412956

Iterative Bounding Box Annotation for Object Detection

Bishwo Adhikari
Tampere University, Finland
bishwo.adhikari@tuni.fi

Heikki Huttunen
Tampere University, Finland
heikki.huttunen@tuni.fi

***Abstract*—Manual annotation of bounding boxes for object detection in digital images is tedious, and time and resource consuming. In this paper, we propose a semi-automatic method for efficient bounding box annotation. The method trains the object detector iteratively on small batches of labeled images and learns to propose bounding boxes for the next batch, after which the human annotator only needs to correct possible errors. We propose an experimental setup for simulating the human actions and use it for comparing different iteration strategies, such as the order in which the data is presented to the annotator. We experiment on our method with three datasets and show that it can reduce the human annotation effort significantly, saving up to 75% of total manual annotation work.**

I. Introduction

Object detection is one of the core research fields in machine learning and computer vision. Recently, object detection algorithms have matured enough to solve real-world vision problems. It serves as the key component in application fields such as face detection [1], pedestrian detection [2], surveillance systems [3], autonomous vehicles [4], [5], etc. The supervised learning principle is widely used in current object detection systems, where a human labeled dataset is used for training the detection model. The performance of supervised machine learning models relies heavily on the amount and quality of annotated training data. However, the challenge in supervised object detection is collecting large, high-quality labeled datasets with the aim of having a well-performing object detection model.

Recently, both the scale and the variety of public datasets for object detection has increased. Among the most popular ones are PASCAL VOC [6], MS COCO [7], OpenImages [8], and Kitti [4] and they are widely used as benchmark datasets in the field of object detection. The labor-intensive and tedious job of object annotation for these large datasets has often been solved by crowdsourcing [9] a large number of human annotators on web platforms such as Amazon Mechanical Turk. However, crowdsourcing may not be a feasible option for annotating small and medium-sized datasets, when the data is confidential in nature, or simply when annotation resources are limited. Hence, there is demand for resource-efficient, user-friendly annotation tools to prepare labeled dataset for machine learning.

Researchers have been mainly focusing on two approaches to reduce the cost of bounding box annotation, weakly-supervised and active learning methods. The weakly-supervised approach uses images and corresponding object labels and lets the network draw bounding boxes. On the other hand the active learning approach trains the model and requests human to draw bounding boxes on a subset of images actively selected by the learner itself. These approaches still require a significant amount of human annotator time for drawing high-quality bounding boxes.

In this paper, we present a simple and practical heuristic to annotate the bounding boxes on image datasets. The iterative annotation approach takes advantage of the trained model to propose labels for a batch of unlabeled images leaving the annotator only for correction work. Thus reducing the work-load of annotation. Compared to other approaches, the iterative approach alternating between training-prediction-annotation balances in the work between machine and annotator for training and correction.

Although commercial and open source tools that learn while annotating do exist (*e.g.,* hasty.ai [10] and ilastic [11]), there is only a limited amount of academic research on the topic. Moreover, our particular focus is not on tool development, but rather a systematic study investigating different *strategies* for an annotation campaign; in particular the order in which the images are presented to the annotator. To this aim, we describe an easy-to-use measure of workload based on precision and recall metrics of the learnt machine learning model.

Our proposed strategy significantly reduces the workload to create environment specific object detection datasets. Iterative training strategy is handy for a data annotation campaign, reduces tedious manual work by assisting annotators in real-time. A single annotator can efficiently annotate whole dataset utilizing a partly trained detector with our iterative training – labels proposal method. We experiment with the *continual learning* [12] effect, an ability of network to learn consecutive tasks without forgetting how to perform on previously trained task, on object detection models in an iterative loop. Additionally, the *catastrophic-forgetting* [12] behaviour of object detection models are experimented with multiple approaches. The term *catastrophic forgetting* resembles neural networks tendency of forgetting knowledge from the former data after learning from the new data.

The rest of this paper is structured as follows. We review the related literature on object annotation in Section II. This is followed by a detailed discussion of the individual components of

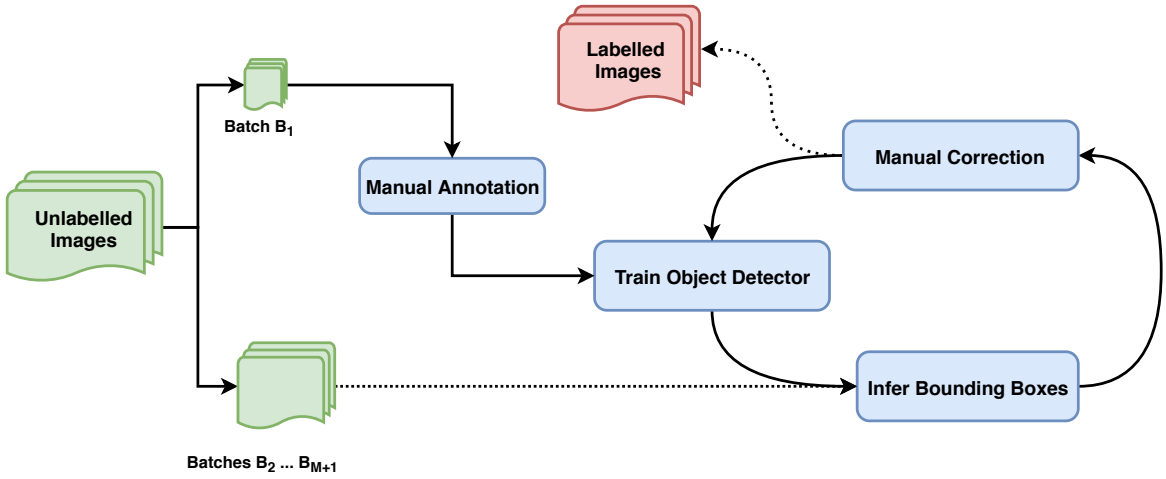


Fig. 1. Our purposed system for iterative bounding box annotation. The object detection model is trained on a small batch of manually annotated images. The trained model is used to predict labels on an unlabeled batch, followed by manual correction. After the first round of *train-infer-correction*, detector is trained on the recently labeled batch. This process continues in a loop until all unlabeled batches are labeled.

the proposed iterative annotation method in Section III. Next, we present our experimental setup with a brief description of the dataset and pre-processing approaches together with the discussion of experimental results in Section IV. Finally, in Section V, we conclude our work and consider potential research directions for the future.

II. Related Work

Object annotation in digital images has been widely studied since the first object detection methods were proposed in computer vision. There have been many studies focusing on speeding up the image dataset annotation for object detection task. In [13], [14], [15], Papadopoulos *et al.* proposed multiple approaches for bounding box annotation. In their bounding box verification approach [13], annotator only needs to verify the label proposed by the network with an accept/reject decision by human. In the click supervision approach [14], the human annotator marks the point in the center of object in an image; and in the extreme clicking approach [15], annotator clicks on four physical points on the object: the top, bottom, left- and right-most points.

Among these works, the box verification approach [13] is the most similar to the proposed one. However, the proposed approach is different from all of these methods in that (1) we treat the object detector as a whole instead of splitting the task into object proposal and classification stages; (2) our human involvement is different (*i.e.,* bounding box *correction* instead of *verification*); (3) we evaluate the performance in terms of total workload; and (4) we also study the order in which the examples are presented to the annotator directing the research towards active learning.

Konyushkova *et al.* proposed learning intelligent dialogs [16] that takes advantage of a trained network to a draw bounding box on image. It requires human annotator to verify the bounding box proposed by the detector in all images. Again, our human interaction model is more straightforward (correction instead if yes/no verification), which in fact simplifies the task and the total workload (as most proposals need no action).

In our previous paper [17], we used a two-stage semi-automatic approach to speed up bounding box annotation on labeling small training dataset and correction of network proposals. The proposed approach is related, but we extend the two-stage approach into an iterative training loop with unlimited number of training iterations rather than just two.

There are commercial annotations tools available, such as hasty.ai [10] and ilastic [11]. Although these tools exist already, the semi-automatic annotation approach *specifically in bounding box annotation* has not been studied in the literature. To the best of our knowledge, this is the first systematic study of how different strategies could work in manual annotations workload reduction.

III. Method

Our focus is in the iterative annotation framework illustrated in Figure 1. This annotation framework uses an incremental learning approach on a small batch of manually labeled images, trains a detection model, uses freshly trained model to propose bounding boxes on a batch of unlabeled images, and requests the annotator do the correction on possible incorrect bounding boxes or labels proposals. The involvement of human annotators is only in the correction stage, hence, decrease the tedious task of manual annotation. Algorithm 1 summaries all steps of the iterative training method. Next, we will describe each component of our method.

A. Manual Annotation

The first step is to fully annotate the first batch of (say, 50) images from the unlabeled dataset. This stage is fully manual and requires human involvement to draw bounding boxes and provide class label on images. We use a basic bounding box annotation tools with no extra speed up procedures. The ways

Algorithm 1: Iterative annotation

Require: Set of unlabeled images split to $M+1$ distinct annotation batches $B_0, \ldots, B_{M+1}$

1: annotate images in batch B_0 manually
2: train object detection model with images from B_0
3: **for** $i \in 1, 2, \ldots, M$ **do**
4: propose annotations for batch B_i using the current prediction model
5: do manual correction for the proposals
6: fine-tune the object detection model with batch B_i
7: **end for**

return fully labeled dataset

of selecting images batch for manual annotation are described later in Section IV-C.

B. Object Detection Model Training

The second step is to train the object detection model. Although any detector can be used, we focus on the recent deep learning-based object detection models. The common practice is to use a pre-trained network and fine-tune on a new dataset [18]. We choose two pre-trained networks trained on the MS COCO [7] dataset and fine-tune on other widely used datasets. Details of our training strategies are explained in Section IV.

C. Bounding Box Proposals

After fine-tuning the object detection model with the batches annotated so far, it is used to predict bounding boxes for the next batch of unlabeled images. The most recently trained detection model proposes bounding boxes and class labels on the next unlabeled batch of images.

D. Manual Correction

The proposed annotations are inspected and manually corrected by a human annotator. The human annotator needs to go through all the proposed labels and bounding boxes. Incorrectly predicted boxes are removed, wrongly labeled classes are corrected and new boxes are drawn, if needed. As the model is presented more samples during the iterations, the human workload should decrease, and the user only needs to accept the boxes in most cases.

E. Estimating the Workload

Our goal is to estimate the human workload in a simulated setting, attempting to find out *how much time a human would spend* on a full annotation campaign. Namely, we can compute the number of corrections required from the user with the commonly used precision and recall metrics. More specifically, the datasets are fully annotated, but we will process them iteratively to measure how much work a human would have done at each stage; and to compare different strategies how this could have been done faster.

The amount of overlap between the ground truth label and the predicted bounding box is used to define the false positives and false negatives. We assume the user would correct the annotation if the *intersection-over-union* (IoU) overlap between the true object location and the predicted bounding box is less than 50%, which is commonly used in performance evaluation in object detection.

The formula to calculate the amount of corrections to bounding boxes and class labels is adopted from Adhikari *et al.* [17]. The number of *additions* the user would have to perform for the next batch B of images is given as

$$\#\text{ additions} = (\#\text{ of true objects}) \times (1 - \text{recall}),$$

with the recall metric computed from ground truth annotations for B. Moreover, the number of removals the user would have to perform (for false positives) is given as

$$\#\text{ removals} = (\#\text{ of all detections}) \times (1 - \text{precision}).$$

with the precision computed from ground truth annotations for B. Finally, the total correction work is the sum of these two steps:

$$\#\text{ corrections} = \#\text{ additions} + \#\text{ removals}.$$

In an ideal case, when the detection model is good enough, the correction work on proposed bounding boxes should take significantly less time than drawing new boxes [17]. Additionally, in the case of partial overlap less than 50%, we model the user operation as removal of the incorrect box and addition of the box at the correct location.

F. Labeled Dataset

After every correction stage, the fully labeled batch is gathered and use for training the detection model for the next iteration. The cycle of training detection model, bounding box inference and correcting on proposed annotations loop continues until all unlabeled images are fully labeled. Hence, the loop produces a fully labeled image dataset for object detection in iterative loop with reduced workload for human annotator.

IV. Experiments and Results

In this section, we evaluate the efficiency of the proposed approach. More specifically, we experiment on the iterative annotation with a number of detection architectures and datasets; both described below.

A. Models

For detection architectures, we study the following two commonly used configurations. Both networks were trained starting with weights pre-trained with MS COCO dataset [7].

***Faster RCNN*—**The faster region convolutional neural network (Faster RCNN) framework proposed by Ren *et al.* [19] is a two-stage detector, where the first stage network creates object proposals, followed by the second stage network classifying the proposals into different categories. For the backbone, we use the 50-layer variant of the residual convolutional network (Resnet50) [20].

SSD—A commonly used lightweight detection architecture, the single shot detector (SSD) framework was proposed by Liu *et al.* [21]. Since the detections are produced directly in a single forward pass of the network, it is often the model of choice for resource limited inference scenarios. For the backbone, we use the Mobilenet V2 [22].

These particular models were selected since Faster RCNN is typically more accurate in detection, while the SSD is faster in inference. As detection model complexity plays crucial role in training, it is always worth to put some effort on model selection for the particular use case. Moreover, the idea is to experiment various strategies to find out an optimal strategy for high quality bounding box annotation efficiently in iterative approach.

B. Datasets

We selected three datasets for our experiments. The intention is to include both large scale datasets (OpenImages) as well as small scale sets (Indoor). The large datasets represent the upper bound in the size of an annotation campaign, while the small scale is the common setup occurring in practice.

PASCAL VOC—PASCAL VOC dataset [6] is a popular benchmark dataset for object detection. The dataset consists of 17k images having 40k object instances from 20 class categories, including person, bus, car, and motorbike. Two sets of independent experiments were conducted in this dataset: first with all classes and second with individual class categories. In the latter case, we used the top ten object categories from VOC 2012 dataset, listed in Table III.

OpenImages—OpenImages [8] is yet another very large dataset for object detection, classification and instance segmentation. The OpenImages V4 contains 9.2M images, 15.4M bounding boxes for 600 object classes. In our experiment, we used a subset of the Open Image Dataset: 10.5k images are selected from the person class; filtering the occluded, truncated, and groups of objects with a single label (multiple objects inside a single bounding box), as these annotations tend to be very noisy.

Indoor—Indoor dataset [17] is a moderate size dataset collected from university indoor premises. The fully annotated object detection dataset consists of about 2,200 images and about 5,000 object instances. Images were extracted from a series of videos. Therefore, this dataset is also associated with a temporal order of samples, which we will exploit later.

Table I summarizes the characteristics of these three datasets. Note that although all datasets used in our experiments are fully labeled, we investigate how their relabeling could benefit from training in the annotation loop. The scope of this research is to experiment how bounding box annotation cloud be done faster with minimal human involvement in an iterative-train loop.

C. Order of annotation

For each iteration, we use a batch size of 50 images in all our experiments. The input image size of 1024x600 pixels is used to train all RCNN models and the input size of 300x300 pixels is used to train all SSD models. The following three annotation orders are experimented on the above mentioned datasets.

Shuffled—All images on the dataset are shuffled in random order and divided into small batches of 50 images. All these small batches have randomly selected images from the whole dataset. The human annotator manually annotates the first batch, then fine-tune the detection model on that batch, and remaining batches are used for annotation proposal in an iterative loop.

Sorted—In this approach images from the dataset are sorted in the decreasing order of number of objects per image. Images having more objects (in both single and multiclass cases) are presented first and so on. In this setup, the first batch of 50 images contains comparatively more objects than the last batch. Hence, it takes more time to annotate the first batch than in any other setups.

Original—In our next approach, images are presented in the original order defined either by the filename (PASCAL VOC and OpenImages) or temporal order in a video (Indoor) or some other inherent way of ordering the examples. This is likely to differ from the shuffled order only for temporal sequences such as video, but we also experiment with this order with the non-temporal datasets for the sake of completeness.

TABLE I
COMPARISON OF INDOOR [17], PASCAL VOC 2012 [6] AND OPENIMAGES [8] DATASETS. SOURCE, SIZE, ANNOTATION FEATURES AND USAGES ARE PRESENTED IN ROWS RESPECTIVELY.

Indoor	**PASCAL VOC**	**OpenImages**
Indoor videos	Online (Flickr)	Online (Flickr)
2.2k images 4.5k instances 7 classes	17.1k images 40k instances 20 classes	9.2 M images 15.4M instances 600 classes
Fully labelled by one annotator; annotations are of high quality	Fully labelled by multiple annotators; annotations are of good quality	Partial labelled multiple annotators & machine generated labels; annotations contain noise
object detection	classification, detection, segmentation	classification, detection, visual relationship

TABLE II
REDUCTION OF MANUAL WORKLOAD (%) WITH DIFFERENT STRATEGIES ON INDOOR, PASCAL VOC 2012 AND OPENIMAGES DATASETS. NUMBERS IN BOLD REPRESENT THE BEST PERFORMING APPROACH ON EACH SECTION.

Network - Approach	**Indoor**	**PASCAL VOC**	**OpenImages Person**
RCNN - Shuffled	**75.86**	18.40	45.62
RCNN - Sorted	56.97	20.93	**60.05**
RCNN - Original	35.78	**25.23**	45.73
SSD - Shuffled	**47.38**	3.46	20.28
SSD - Sorted	31.58	5.66	**35.13**
SSD - Original	19.24	**7.97**	20.04

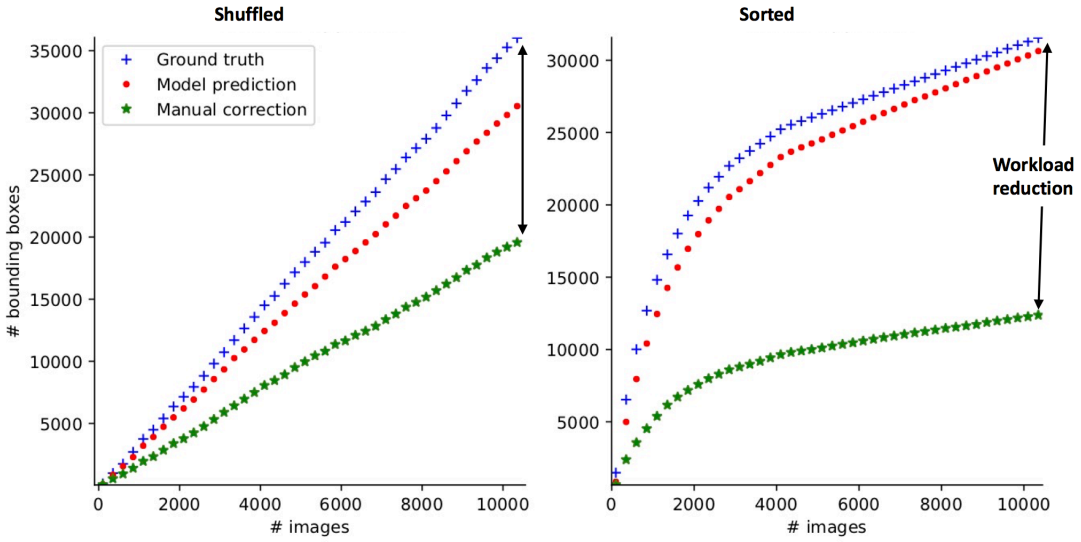


Fig. 2. An example of the effect of the order of iterative annotation. The figures show the cumulative number of ground truth boxes, boxes predicted by the RCNN model, and the manual corrections required on the OpenImages/Person dataset. Images are annotated in a random order (left) and in an order sorted by the # of boxes per image (right). The reduction in workload is significantly better in the sorted order (right).

TABLE III
COMPARISON OF THE MANUAL WORKLOAD REDUCTION (%) ON INDIVIDUAL CLASS CATEGORIES FROM PASCAL VOC 2012 DATASET.

	Airplane	**Bird**	**Boat**	**Bottle**	**Car**	**Cat**	**Chair**	**Dog**	**Person**	**Plant**	**Average**
RCNN - Shuffled	56.14	50.30	**35.70**	44.49	51.96	55.34	29.31	57.87	44.61	**38.72**	46.44
RCNN - Sorted	**62.07**	**60.43**	35.65	**46.68**	**56.27**	59.53	**32.44**	63.28	**61.24**	32.75	**51.03**
RCNN - Original	53.87	50.41	32.50	41.54	55.14	**61.58**	29.30	**61.38**	57.16	34.64	47.75

D. Results

An example of the progression of our method experimented on OpenImages person class using the RCNN detection model is shown in Figure 2. The amount of ground truth, model prediction, and manual correction in terms of numbers of bounding boxes as a function of the number of images are presented with shuffled and sorted approaches.

The left graph shows the experimental result based on the shuffled order of images from the dataset, and the right graph shows the experimental result based on the sorted order. The manual workload is less in the sorted order approach; only 13,414 manual corrections would be needed instead of 33,573 ground truth bounding boxes. In the case of the random shuffle approach, 19,938 manual corrections would be required instead of 36,659 ground truth bounding boxes.

The higher the gap between ground truth and manual correction, the better the annotation performance in terms of manual workload reduction. The first stage manually annotated bounding boxes from the first batch (B_0) of images are not included in these graphs. For the sorted approach, the annotation workload for the very first batch is usually more than other approaches.

The reductions of manual workload required to correct the proposed bounding boxes in experimented approaches are shown in Table II. The prediction performances of the two-stage RCNN model are comparatively better than those of the single-stage SSD model. The RCNN model capacity and higher input resolution for training images result in good proposals for bounding boxes and class labels. However, the two-stage models are computationally expensive compared to single-stage detection models like SSD. Moreover, the results follow the same pattern for both methods in all of our experimented approaches. The minimum amount of manual workload reduction noted is 3.46 % with the SSD model on the VOC multiclass dataset with the shuffling approach. On the other hand, the maximum amount of reduction recorded is 75.86% with the RCNN model on the Indoor dataset with the shuffling approach.

Interestingly, it is seen that shuffling images helps to improve the overall performance of the detection model only in the case where images are continuous frames of video (Indoor). As shown in Table II, with the RCNN model, 75.86% of manual work required to draw bounding boxes can be reduced by randomly shuffling the images from the indoor dataset. However, in the shuffling approach, there is a high probability of having every next image from the sequence in different batches. The catastrophic forgetting effect seems to be least in this approach hence, providing accurate proposals for bounding boxes and class labels. Moreover, the results on OpenImages and VOC datasets show that the shuffled approach is the worst performing among the compared approaches.

For multi-class datasets, the annotations can be done one class at a time (iterating over all images for each class) or all classes simultaneously. Therefore, we experimented with the ten most populated classes of the VOC dataset with the RCNN detection model, with a single category fully annotated iteratively at a time. The results are shown in Table III. Interestingly, the workload reduction is doubled when annotations are done one class at a time (average reduction up to 51.03%) compared to the multi-class iterative annotation (reduction 25.23%). In terms of individual categories, the reduction ranges from 32.4% (chair) to 62.1% (airplane). The likely reason for the improved performance is that the detection model does not have to learn away from the other categories and can focus on learning a single class at a time. Obviously, this approach requires that images are initially sorted by category: Otherwise we would unnecessarily present, *e.g.,* non-airplane images while annotating the airplane class.

Additionally, it is found that the sorted approach is optimal for the manual workload reduction in most of the single class categories. On the OpenImages person class, as shown in Table II, the workload reduction with the sorted approach is higher than other setups. The reduction in manual workload with RCNN model is 60.05% and with SSD model is 35.13%. Most of the larger reductions in Table III come from the sorted setup.

On the other hand, by splitting images into different batches based on the filename or temporal order gives the best result for the multiclass dataset. The original order approach gives best performance for VOC multiclass and second best performance for VOC and OpenImages single class datasets.

TABLE IV
COMPARISON OF THE MANUAL WORKLOAD REDUCTION (%) WITH ITERATIVE ANNOTATION AND TWO-STAGE METHOD ON INDOOR DATASET [17]. ALL EXPERIMENTS ARE DONE WITH FASTER RCNN RESNET101 AND 0.5 IOU THRESHOLD IS USED.

Approach	**Reduction (%)**
Two-stage (5%) [17]	79.47
Two-stage (6%) [17]	81.21
Two-stage (8%) [17]	78.68
Two-stage (10%) [17]	79.03
Two-stage (20%) [17]	72.46
Ours (iterative)	79.56
Ours (cumulative)	80.56

We performed some more experiments on the Indoor dataset to have a comparison with the previous method [17]. In [17], a two-stage approach was used, with first fold annotated fully manually, and the second part using the proposals of a model trained with the first fold. The proportion of samples in the first and second folds is a tuning parameter.

Table IV shows the results of the two-stage approach [17] for selected split ratios; with 6% for the first fold and 94% for the second fold reducing the workload the most (81.21%). Using the fully iterative approach of this paper results in a comparable accuracy (79.56%), but does not require the selection of the split ratio.

However, the sweet spot of 6% is the *only* split ratio exceeding the workload reduction of the proposed iterative approach. In fact, it is impossible to know the optimal split ratio *a priori* for an unlabeled dataset; for example, the 5% – 95% split is already almost 2% worse. Thus, our iterative method is economical compared to the earlier two-stage approach.

In addition to the iterative approach, we experimented on a modified version of the proposed method. Namely, instead of training on the most recent batch, one can train with all previously annotated batches at each iteration. This *cumulative* approach was tested on the Indoor dataset and the results are shown on the last line of Table IV. It turns out that the cumulative approach is better compared to the iterative approach; however, its training becomes significantly slower due to larger amount of training data.

In general, the two-stage approach with other amounts of manually labeled images has less workload reduction than our iterative approach trained with the same approach and same model. The significant benefit iterative approach over this is that the model is improving over time; object proposals are getting better, hence require less correction work. Also, labeling a small batch of images is more relaxed and less error-prone compared to mass annotation.

V. CONCLUSION

This paper presented an iterative human-in-the-loop annotation method to minimize human involvement in image annotation campaigns for the task of object detection. The proposed method utilizes human-machine collaboration resulting in high-quality bounding box annotations on small and medium-sized datasets. Extensive experiments on the proposed method on three datasets show that our iterative annotation is able to reduce the manual workload by up to 75%. However, the reduction of manual annotation workload appears to be dependent on the dataset size, image source, and object class categories. We also noted an adequate performance of proposed approaches on the already existing platform for the bounding box annotation campaign.

Our implementation uses a commonly used threshold value 0.5 IoU threshold threshold for detection performance measurement and accepting the label proposals. In our future work, we will consider the effect of requiring a higher IoU for the annotation results (> 0.5 IoU). In fact, we already experimented with a 0.8 IoU threshold, and discovered that the manual workload reduction is even higher since less manual correction work is needed on the good quality proposals. It would also be interesting to experiment on the effect of an IoU threshold on the network proposals and its tradeoff between correction workload. The resource constraints and latency of the detection model could be studied in the future with parameters such as batch size, training step, and input size of images for the model training.

Furthermore, it would be nice to experiment on an active learning pipeline that could reduce human workload even more by automatically selecting informative images for labeling on earlier iterations.

ACKNOWLEDGMENT

This work was financially supported by the Business Finland project *408/31/2018 MIDAS*.

REFERENCES

[1] E. Hjelmås and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, 2001.

[2] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, "Real-time pedestrian detection with deep network cascades," in *BMVC*, 2015.

[3] A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya, "Object detection algorithms for video surveillance applications," *ICCSP*, 2018.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[5] S. Chadwick, W. Maddern, and P. Newman, "Distant vehicle detection using radar and vision," *arXiv preprint arXiv:1901.10951*, 2019.

[6] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, 2015.

[7] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *Computing Research Repository*, vol. abs/1405.0312, 2014.

[8] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[9] H. Su, J. Deng, and L. Fei-fei, "Crowdsourcing Annotations for Visual Object Detection," in *AAAI Human Computation Workshop*, 2012, pp. 40–46.

[10] "Hasty," https://hasty.ai/, 2020.

[11] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J. I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F. A. Hamprecht, and A. Kreshuk, "ilastik: interactive machine learning for (bio)image analysis," *Nature Methods*, Sep. 2019.

[12] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *arXiv preprint arXiv:1612.00796*, 2016.

[13] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," in *CVPR*, 2016, pp. 854–863.

[14] ——, "Training object class detectors with click supervision," in *CVPR*, 2017, pp. 180–189.

[15] ——, "Extreme clicking for efficient object annotation," in *ICCV*, 2017, pp. 4940–4949.

[16] K. Konyushkova, J. R. R. Uijlings, C. H. Lampert, and V. Ferrari, "Learning intelligent dialogs for bounding box annotation," in *CVPR*, 2018, pp. 9175–9184.

[17] B. Adhikari, J. Peltomaki, J. Puura, and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," in *7th European Workshop on Visual Information Processing (EUVIP)*, 2018.

[18] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, Oct 2010.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *European conference on computer vision*, 2016, pp. 21–37.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

PUBLICATION

III

Sample Selection for Efficient Image Annotation

B. Adhikari, E. Rahtu and H. Huttunen

European Workshop on Visual Information Processing (EUVIP), 2021
DOI: 10.1109/EUVIP50544.2021.9484022

SAMPLE SELECTION FOR EFFICIENT IMAGE ANNOTATION

Bishwo Adhikari, Esa Rahtu

Tampere University, Finland

Heikki Huttunen

Visy Oy, Finland

ABSTRACT

Supervised object detection has been proven to be successful in many benchmark datasets achieving human-level performances. However, acquiring a large amount of labeled image samples for supervised detection training is tedious, time-consuming, and costly. In this paper, we propose an efficient image selection approach that samples the most informative images from the unlabeled dataset and utilizes human-machine collaboration in an iterative train-annotate loop. Image features are extracted by the CNN network followed by the similarity score calculation, Euclidean distance. Unlabeled images are then sampled into different approaches based on the similarity score. The proposed approach is straightforward, simple and sampling takes place prior to the network training. Experiments on datasets show that our method can reduce up to 80% of manual annotation workload, compared to full manual labeling setting, and performs better than random sampling.

***Index Terms*—** Object Detection, Image Annotation, Sample Selection, Bounding Box

1. INTRODUCTION

Object detection is one of the fundamental and widely studied problems in computer vision. The success of convolutional neural networks (CNN) and deep learning has transformed the domain of object detection. These approaches outperform earlier techniques by a large margin, but still behind the human-level understanding. Recently, object detection has been mature enough to be used in applications in various domains such as agriculture [1], medical imaging [2], robotics [3], and remote sensing [4].

Supervised object detection is the most widely used approach but requires a large amount of labeled examples for the training, with the labels usually assigned by human annotators. In object detection, the datasets consist of images, and each image can have multiple annotations; the acquisition of a large number of high-quality datasets is tedious, time-consuming, costly, and often requires expertise (e.g. medical images). Object detection breakthroughs in various fields have been facilitated by a large number of annotated benchmark datasets available from multiple domains [5, 6, 7]. Object detection tasks can be very specific to a certain environment and condition, thus relying only on public datasets often does not generalize well.

Several existing works have been reported on methods for data-efficient object detection training including transfer learning [8], semi-supervised learning [9], and weakly supervised learning [10]. However, most of these methods still require a certain amount of domain-specific labeled training data. Crowd-sourcing [11] have been increasingly popular due to the progress in third-party services such as Amazon Mechanical Turk (AMT) for data annotation. As there is no standard to measure the quality of the annotations, the annotated datasets via these platforms often have noisy labels *i.e.*, missing labels, inaccurate labels, and only part of the object is being labeled. Moreover, not all projects can opt for these solutions due to the privacy concern and cost of high-quality annotation from expert annotators. Active learning [12] is another popular technique that aims to reduce the training time by actively querying for the labels of unlabeled instances. Recently, active learning techniques have been commonly used for the image classification task but not commonly available for the object detection task. Hence, there is a need for faster and efficient methods for labeling image examples for object detection model training.

In this work, we focus on reducing manual annotation workload using active image sampling. Our method selects and sorts informative images into mini-batches that will lead to less annotation work; utilizing the human-machine collaboration in the continuous train-annotate loop. We use a distance-based image sampling based on the entire image-to-image pairwise Euclidean distance. We propose two sampling approaches that can reduce the overall annotation workload. Secondly, we utilize a self-supervised network for the feature extraction. Experiments on three datasets show that the proposed method significantly reduces the annotation workload compared to the full human-level manual annotation. The saving in annotation cost is up to 80% in the best case and above 50% on average.

The rest of the paper is structured as follows. The literature review of related works is presented in Section 2. In section 3, we present a detailed description of our method and its components. In Section 4, details about the experimental setups, dataset, and results are presented and discussed. Finally, we conclude this paper with our findings in Section 5.

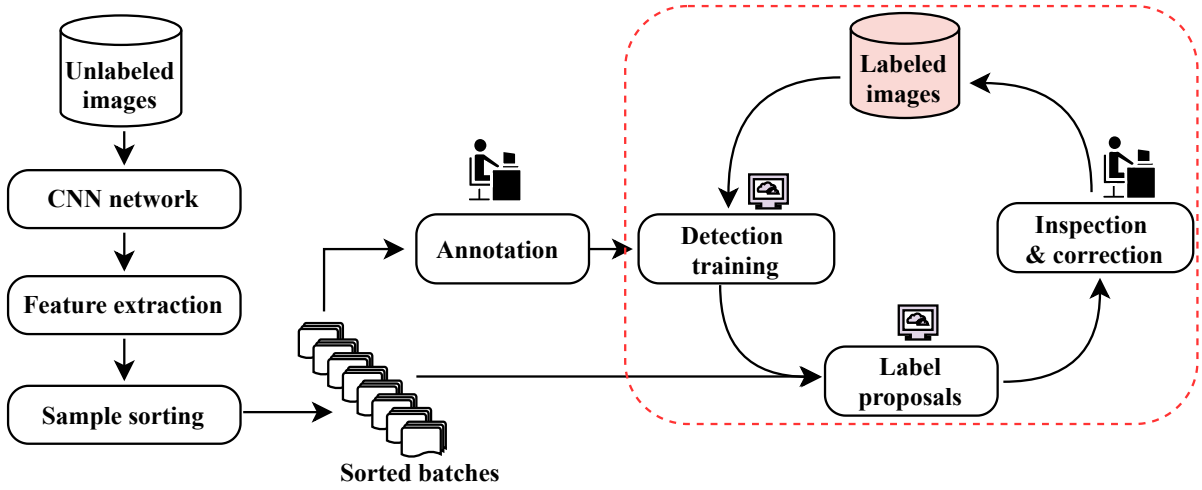


Fig. 1. Proposed efficient annotation scheme. The sample selection is done before the object detection model training. The train-annotation loop (inside red) continues till all unlabeled batches are taken and labeled.

2. RELATED WORK

2.1. Active Learning

Active learning [12] aims to reduce the labeling cost by selecting the most useful samples from the unlabeled dataset and request for the labels to maximize a model performance while trained with these labels. Active learning has been proven to be effective in reducing annotation cost on image classification task [13].

For object detection, uncertainty-based sampling is commonly used as a measure of informativeness [14, 15]. The key principle of this approach is to train a learner, calculate the degree of uncertainty, and then query an unlabeled instance with the least confidence. The drawback of uncertainty-based approaches is that they rely on a labeled dataset to build an initial model to select the query instance, and the performance is often unstable when there are only a few labeled samples available. Querying every unlabeled sample using a CNN network is computationally expensive. In [16], a feature descriptor function was proposed to generate features for all images and the most distant sample is selected based on the euclidean distance measure from these features. This is feasible for a single-class dataset but requires heavy computation since every sampling instance requires a new round of training and evaluation with the CNN model.

Our method takes all unlabeled images for the sampling prior to network training, representing all samples from the unlabeled dataset. Furthermore, it does not rely on the prediction of classification scores, hence it is efficient and straightforward. The proposed sample selection method is close to [16] but in this work we sampled all images based on the distance metric before the network training, resulting in a computationally efficient workflow.

2.2. Self-training

Many previous works have used a trained model to predict labels for a set of unlabeled images and involve humans in the process of correcting predicted labels [2, 17, 18, 19, 20]. In [18], a two-stage method proposed for speeding up bounding box annotation; model training on a small set of a labeled dataset, inference labels for the unlabeled set followed by manual correction on network proposals. In [20], an iterative train-annotate loop is proposed for efficient image annotation in a small batch of images at a time. This method explicitly used the information from ground truth for the selection of images on three approaches. In [19], assistive learning feedback loop is proposed that utilizes the contextual sampling criteria; uniqueness, and average euclidean distance of the images. However, in reality, often such information is not available for the unlabeled dataset. In this work, we do not need such information from unlabeled samples.

The major difference from all these works is that (1) our method does not require explicit information from the unlabeled images; (2) all the sampling task is done before the training phase; (3) we use an entire image for the sample selection based on the extracted features; and (4) human annotator has a rather easy task of labels inspection and/or correction that does not need prior machine learning skills. Among these works, the iterative annotation approach [20] is the most similar to the proposed one.

3. METHOD

The proposed method applies CNN based feature extractor to sample the images in mini-batches based on the distance metric *i.e.,* most similar or most distant images together. Human annotator labels first batch (B_1), trains object detection model

(M_1) on this batch B_1. The freshly trained model M_1 is used to predict bounding boxes and class labels on the unlabeled batch (B_2). The predicted bounding boxes and labels are then inspected and corrected by a human annotator resulting in a fully labeled batch B_2. The next iteration starts with training model M_2 on a combination of labeled B_1 and B_2 and predicts labels for the batch B_3. The train-prediction-correction loop continues till the last batch B_n of unlabeled images is labeled. A human annotator is required for the manual labeling of B_1 and in the inspection and correction stage, inside the loop, as shown in Figure 1.

3.1. Feature Extraction

The first step in the proposed method is to extract image features from all images in the unlabeled dataset. For the feature extraction, we experiment with two networks; ResNet50 [21] network trained on ImageNet [22] (ImgNet) and similarity network (SimNet) trained on images from the unlabeled dataset. Moreover, any CNN network can be utilized as a feature extraction network. SimNet uses ResNet50 as a backbone and structured lifted loss [23] as loss function. Images from the unlabeled dataset are horizontally split into pairs, and the SimNet is trained on these pairs for 25 epochs. Experiments showed that features extracted from this network perform better in sample selection *i.e.,* mapping similar and dissimilar samples.

3.2. Image Sampling

The extracted image features are used to compute the pairwise Euclidean distances of each pair. The images are then sampled based on the distance measure and sorted into batches, later used in the iterative train-annotate loop. We have experimented with three strategies for batch selection; similar (images with the least distance are placed together), dissimilar (images with the maximum distance are placed together), and random sampling (images are randomly sampled into batches). Additionally, we experiment on temporal order if applicable to the dataset.

We start with randomly selecting few images as a query list. On a dissimilar approach, an image that has the largest distance compared to all images in the query list is appended to the query list one at a time, the process continues till all images are added to the list, resulting in distinct images together. While, on a similar approach, an image that has the least distance compared to all images in the query list is added to the list one at a time till all images are ordered in the list, resulting in similar images together.

3.3. Training, Inference and Correction

At first, the detection network is trained on the manually labeled first batch B_1. After the first round, it is trained on all labeled sets available at that time. The process continues till all unlabeled samples are being used in the proposal stage or the desired amount of data is labeled. The number of iterations and other hyperparameters can be decided depending on the available annotation budget.

The freshly trained model is used to infer bounding box and class labels on the batch of unlabeled images. After this, the task of inspection and correction on these proposals is assigned to a human annotator. This task includes adding missing bounding boxes and labels, correcting incorrectly drawn boxes and labels, and removing extra boxes and labels. As the model gets better, the more tedious task is done by the machine, thus simplifying the task of a human annotator.

3.4. Workload Calculation

We follow the previous works [18, 20, 14] in formulating the annotation workload calculation. However, unlike [20], the annotation costs include the workload required to label images from the first batch $B1$, and costs are assumed to be different for different actions during the correction stage.

The annotation workload in terms of bounding boxes is given as

$$\text{Workload } (W_B) = \text{\# corrections} + \text{\# of object in } B_1$$

where # corrections represents the sum of additions (A_n) and removals (R_n) from the manual correction stage. Simply counting the number of added and removed bounding boxes.

On average, these addition and removal actions, required at the correction stage, have different levels of complexity. Hence, the number of objects (bounding boxes) does not provide a realistic estimate of the entire dataset labeling cost. Therefore, we formulate the labeling cost in terms of time (sec.) as

$$\text{Workload } (W_T) \;=\; B_{1n} \times T + A_n \times T + R_n \times T/2$$

where B_{1n}, A_n, and R_n are the number of bounding boxes in B_1 (labeled by the annotator at the beginning), manual additions and removals respectively. While T is the time to annotate one object from scratch.

In [18], the annotation workload calculation uses only the manual correction stage. However, in practice, taking into account all manual work required from all stages would give a better estimation of the annotation workload. The time estimation to draw bounding boxes is conservative and there have been various estimations presented in [17, 18, 19]. In our work, we calculate the annotation cost reduction (%) compared to what it would have taken in a full manual annotation.

4. EXPERIMENTS

4.1. Datasets

PASCAL VOC — PASCAL VOC [5] is one of the popular datasets in vision research. The object detection version consists of fully labeled in 20 object classes including animals,

vehicles, and common household objects. In our experiments, we use all 9963 images with 30638 object instances from both trainval and test sets of the 2007 version. In this dataset, we performed two sets of experiments: first with all class categories and second with individual class categories.

KITTI **—** KITTI [6] object detection and object orientation estimation benchmark consist of 7481 training images and 7518 test images, comprising a total of 80256 labeled objects. The dataset is collected in five scenarios: city, residential, road, campus, and person. We use the train split from object detection category consisting of 7481 images with 40570 object instances of 8 class categories including vehicles (car, van, tram, truck), and person involved in different actions (pedestrian, sitting, cyclist).

Indoor **—** Indoor dataset [18] is a moderate size dataset collected from indoor premises. This dataset has 2213 images and about 4595 object instances of 7 indoor scene classes. The dataset consists of safety signs (exit, fire extinguisher), furniture(trash bin, chair), and equipment (clock, printer, screen). Images were extracted from a series of videos preserving temporal order.

4.2. Implementation Details

For object detection, we experiment on two commonly used detection networks: single-stage SSD [24] with MobileNetV2 [25] backbone and two-stage Faster RCNN [26] with ResNet50 backbone. Both networks are fine-tuned with pre-trained weight from COCO dataset [7]. We select these networks following their popularity and speed vs accuracy trade-off reported in the literature. The SSD network is lighter and faster but less accurate than the alternatives. However, it is a common choice for resource-limited scenarios. On the other hand, Faster RCNN is more complex, computationally heavy but optimal for the high-quality annotation proposals. We resized images to 300×300 pixels for the SSD network and 600×1024 pixels for the Faster RCNN network, as reported on their original papers.

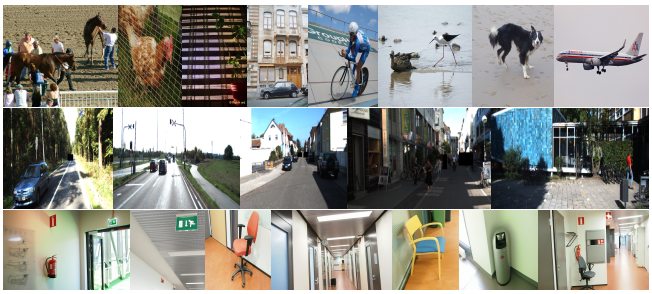


Fig. 2. Example samples selected using the dissimilar approach. Images from PASCAL VOC, KITTI, and Indoor datasets from top to bottom rows, respectively.

We mainly experiment on three different sampling strategies: similar, dissimilar, and random shuffle, mentioned in 3.2. An additional experiment is done with temporal sampling on the Indoor dataset. In Figure 2, we show images sampled by our dissimilar approach in three datasets.

Table 1. Reduction of manual annotation time W_T (%), higher the better, with different approaches. The result is an average of five independent runs. The best for each dataset per column is in bold font.

Dataset	Approach	RCNN		SSD
		0.5 IoU	0.7 IoU	0.5 IoU
Pascal VOC	Shuffle	56.00	55.37	31.61
	Sim (ImgNet)	56.05	54.71	**32.97**
	Dis (ImgNet)	56.48	55.63	32.08
	Sim (SimNet)	55.27	53.65	30.53
	Dis (SimNet)	**56.82**	**56.02**	32.08
KITTI	Shuffle	50.82	49.53	32.21
	Sim (ImgNet)	37.31	34.68	19.92
	Dis (ImgNet)	**51.49**	**50.59**	**33.61**
	Sim (SimNet)	46.67	43.34	28.64
	Dis (SimNet)	49.81	47.16	28.89
Indoor	Shuffle	81.20	80.37	65.98
	Temporal	37.47	35.99	20.24
	Sim (ImgNet)	59.12	59.47	43.35
	Dis (ImgNet)	**81.83**	78.81	64.09
	Sim (SimNet)	67.06	67.38	48.32
	Dis (SimNet)	79.08	**81.64**	**67.76**

Table 2. Reduction of manual workload (%) in terms of bounding boxes W_B with different approaches. The result is an average of five independent runs. The best for each dataset per column is in bold font.

Dataset	Approach	RCNN		SSD
		0.5 IoU	0.7 IoU	0.5 IoU
Pascal VOC	Shuffle	**43.69**	47.11	26.17
	Sim (ImgNet)	43.64	47.18	**27.50**
	Dis (ImgNet)	43.12	47.39	26.46
	Sim (SimNet)	43.40	46.22	26.15
	Dis (SimNet)	43.66	**48.22**	2.15
KITTI	Shuffle	45.93	45.59	27.90
	Sim (ImgNet)	28.08	28.12	14.49
	Dis (ImgNet)	**44.39**	**46.33**	**29.37**
	Sim (SimNet)	39.24	38.49	23.89
	Dis (SimNet)	42.58	42.57	25.03
Indoor	Shuffle	77.11	78.19	60.83
	Temporal	34.77	34.29	17.60
	Sim (ImgNet)	56.10	57.25	39.43
	Dis (ImgNet)	**78.10**	76.51	58.57
	Sim (SimNet)	63.17	65.07	43.25
	Dis (SimNet)	74.08	**78.43**	**63.70**

4.3. Results

We compare the performance of the proposed sampling approaches for labeling the full dataset in a simulated envi-

Table 3. Annotation workload reduction(%) in time (W_T) and bounding boxes (W_B) on individual class categories of the PASCAL VOC 2007 dataset. The best result per category is in bold font.

Approach	Plane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Bike	Person	Plant	Sheep	Sofa	Train	TV	Average
Dis SimNet (W_T)	41.12	**57.00**	44.12	**24.39**	40.97	**41.30**	67.30	58.82	**48.25**	**41.38**	37.35	62.68	**58.80**	**58.98**	74.59	**38.78**	28.39	45.13	**52.69**	44.71	48.34
Dis SimNet (W_B)	30.21	**47.71**	34.97	**18.71**	36.94	**29.18**	59.15	46.11	**37.91**	**36.05**	11.49	49.71	**51.06**	**49.01**	66.54	**29.91**	25.30	17.54	**41.58**	36.53	37.78
Shuffle (W_T)	**46.80**	53.22	**46.93**	23.51	**45.66**	40.58	**68.74**	**60.80**	47.71	40.95	**41.70**	**64.27**	54.74	51.91	**75.54**	37.83	**29.21**	**45.49**	49.44	**47.94**	**48.65**
Shuffle (W_B)	**39.40**	45.10	**39.57**	16.68	**40.58**	27.85	**59.65**	**47.43**	37.24	37.51	**25.78**	**50.37**	45.07	44.79	**67.53**	29.98	**26.65**	**22.17**	36.98	**38.59**	**38.95**

ronment. The amount of workload reductions for different datasets with different sampling approaches and two detection networks is shown in Table 1. Additionally, we experimented on two IoU thresholds; 0.5 and 0.7 which are commonly used as performance measurement in object detection. In all experiments, our sampling approaches perform better in saving manual annotation time. The Faster RCNN model with dissimilar sampling performs better in most cases. The difference between the two IoU thresholds is surprisingly small. We believe this is due to our calculation strategy. In an ideal case, with 0.5 IoU, the network proposes too many labels which the annotator needs to remove in the correction stage. While with 0.7 IoU, the network purposes fewer tight boxes where the annotator needs to add some labels manually, at the correction stage.

In [20], the workload reduction is calculated in terms of the bounding boxes. We show the saving in terms of bounding boxes in Table 2. However, we believe that saving in terms of time would be more practical. Next, we experiment on all categories of PASCAL VOC 2007 taking one class at a time, shown in Table 3. We trained a Faster RCNN network with two approaches: random shuffle and dissimilar sampling order based on our similarity network.

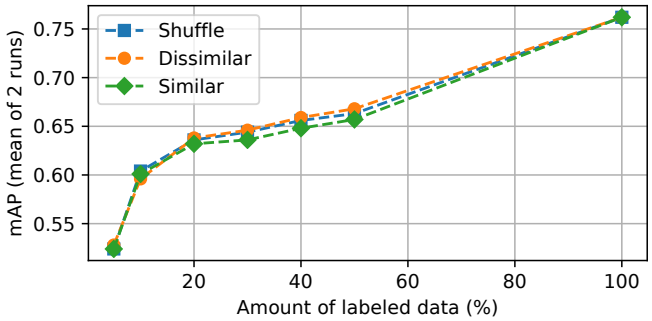


Fig. 3. Relationship between mean average precision (mAP) and amount of labeled dataset. Faster RCNN network is trained on PASCAL VOC 2007 *all* with three sampling approaches and tested with PASCAL VOC 2012 *trainval*.

Next, we experiment with the relationship between the amount of sampled data with the model performance on the PASCAL VOC 2012 *trainval* set. The result in Figure 3 indicates that the proposed methods improve the performance of the object detector when there are fewer labeled samples available for training. As the amount of labeled data increases the performance of the model given all three approaches converges.

4.4. Discussion

The reduction in workload varies among datasets. We believe this is due to the nature of datasets (e.g. type, size, and ground truth). For the Indoor dataset, which is extracted from video sequences and sparsely labeled, the saving is higher. For the KITTI dataset which is densely labeled, the saving is lower due to dense annotations. While in the PASCAL VOC dataset, which is collected from the web, manual workload reduction with our method is moderate. Since the dataset is already so diverse, there is less to achieve with sample selection.

Furthermore, the results obtained on a single class states that the annotation workload reduction with our method is higher for difficult classes. We get better results on classes such as potted plant, chair, and boat that are considered to be hard [27]. These difficult classes usually appear in complex contexts with many objects, occlusion, and varying illumination. In the case of a large-scale dataset even 1% of workload reduction has a significant impact. For example, in the PASCAL VOC, 1% of 30680 is 306 object instances. According to [11], labeling this many objects takes about 3 hours.

The quality of labels obtained from the self-training method is equally important as the labeling cost. Since a human annotator is used to inspect and correct each object proposal in our method, the quality of the labeled dataset is good compared to other semi-supervised approaches. Experiment results with different IoU thresholds show that the proposed method is effective to get tighter bounding boxes, with 0.7 IoU threshold, the workload reduction in annotation time is higher than that of 0.5. This concludes that the proposed method is feasible for efficient sample selection for the cost-effective annotation campaign.

5. CONCLUSION

We proposed a similarity-based self-trained approach for efficient labeling of object detection datasets. The proposed approach can produce high-quality annotation with a reasonable annotation budget. Most of the tedious work is done by the machine while the human annotator mostly takes care of correction work which is often easier than labeling images from scratch. Extensive experiments on three datasets showed that a large amount of manual annotation work can be saved if some focus is paid on sample selection prior to the network training. In the future, we would like to apply this method to more challenging video datasets annotation and semantic annotation.

6. REFERENCES

[1] Mikkel Kragh, Rasmus N. Jørgensen, and Henrik Pedersen, "Object detection and terrain classification in agricultural fields using 3d lidar data," in *Computer Vision Systems*. 2015, pp. 188–197, Springer.

[2] Brendon Lutnick, Brandon Ginley, Darshana Govind, Sean D. McGarry, Peter S. LaViolette, Rabi Yacoub, Sanjay Jain, John E. Tomaszewski, Kuang-Yu Jen, and Pinaki Sarder, "An integrated iterative annotation technique for easing neural network training in medical image analysis," *Nature Machine Intelligence*, vol. 1, no. 2, pp. 112–119, Feb 2019.

[3] Takuya Kiyokawa, Keita Tomochika, Jun Takamatsu, and Tsukasa Ogasawara, "Fully automated annotation with noise-masked visual markers for deep-learning-based object detection," *IEEE Robotics and Automation Letters*, vol. 4, 2019.

[4] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 296–307, 2020.

[5] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, 2008.

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[7] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft COCO: common objects in context," *Computing Research Repository*, vol. abs/1405.0312, 2014.

[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, Oct 2010.

[9] L. Beyer, X. Zhai, A. Oliver, and A. Kolesnikov, "S4l: Self-supervised semi-supervised learning," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1476–1485.

[10] Zhi-Hua Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2017.

[11] Hao Su, Jia Deng, and Li Fei-fei, "Crowdsourcing Annotations for Visual Object Detection," in *AAAI Human Computation Workshop*, 2012, pp. 40–46.

[12] Burr Settles, "Active learning literature survey," Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[13] X. Li and Y. Guo, "Adaptive active learning for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 859–866.

[14] Sai Vikas Desai, Akshay L. Chandra, Wei Guo, Seishi Ninomiya, and Vineeth N. Balasubramanian, "An adaptive supervision framework for active learning in object detection," in *BMVC*, 2019.

[15] Donggeun Yoo and In So Kweon, "Learning loss for active learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Asim Smailagic, Hae Young Noh, Pedro Costa, Devesh Walawalkar, Kartik Khandelwal, Mostafa Mirshekari, Jonathon Fagert, Adrian Galdran, and Susu Xu, "Medal: Deep active learning sampling method for medical image analysis," *arXiv preprint arXiv:1809.09287*, 2018.

[17] Ksenia Konyushkova, Jasper R. R. Uijlings, Christoph H. Lampert, and Vittorio Ferrari, "Learning intelligent dialogs for bounding box annotation," in *CVPR*, 2018, pp. 9175–9184.

[18] B. Adhikari, J. Peltomaki, J. Puura, and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," in *7th European Workshop on Visual Information Processing (EUVIP)*, 2018.

[19] V. W. H. Wong, M. Ferguson, K. H. Law, and Y. T. Lee, "An assistive learning workflow on annotating images for object detection," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1962–1970.

[20] B. Adhikari and H. Huttunen, "Iterative bounding box annotation for object detection," in *25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 4040–4046.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[23] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese, "Deep metric learning via lifted structured feature embedding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[27] Radu Tudor Ionescu, Bogdan Alexe, Marius Leordeanu, Marius Popescu, Dim Papadopoulos, and Vittorio Ferrari, "How hard can it be? Estimating the difficulty of visual search in an image," in *Proceedings of CVPR*, June 2016, pp. 2157–2166.

PUBLICATION
IV

Effect of Label Noise on Robustness of Deep Neural Network Object Detectors

B. Adhikari, J. Peltomäki, S. B. Germi, E. Rahtu and H. Huttunen

Computer Safety, Reliability, and Security. SAFECOMP Workshops. Ed. by I. Habli, M. Sujan, S. Gerasimou, E. Schoitsch and F. Bitsch, 2021, 239–250
DOI: 10.1007/978-3-030-83906-2_19

Effect of Label Noise on Robustness of Deep Neural Network Object Detectors

Bishwo Adhikari[1][0000−0003−0037−8313],
Jukka Peltomäki[1][0000−0002−9779−6804],
Saeed Bakhshi Germi[1][0000−0003−3048−220X],
Esa Rahtu[1][0000−0001−8767−0864], and
Heikki Huttunen[2][0000−0002−6571−0797]

[1] Tampere University, Tampere, Finland
[2] Visy Oy, Tampere, Finland
{bishwo.adhikari, jukka.peltomaki, saeed.bakhshigermi, esa.rahtu}@tuni.fi
heikki.huttunen@visy.fi

Abstract. Label noise is a primary point of interest for safety concerns in previous works as it affects the robustness of a machine learning system by a considerable amount. This paper studies the sensitivity of object detection loss functions to label noise in bounding box detection tasks. Although label noise has been widely studied in the classification context, less attention is paid to its effect on object detection. We characterize different types of label noise and concentrate on the most common type of annotation error, which is missing labels. We simulate missing labels by deliberately removing bounding boxes at training time and study its effect on different deep learning object detection architectures and their loss functions. Our primary focus is on comparing two particular loss functions: cross-entropy loss and focal loss. We also experiment on the effect of different focal loss hyperparameter values with varying amounts of noise in the datasets and discover that even up to 50% missing labels can be tolerated with an appropriate selection of hyperparameters. The results suggest that focal loss is more sensitive to label noise, but increasing the gamma value can boost its robustness.

Keywords: Safe AI · Deep Neural Networks · Label Noise · Image Labeling

1 Introduction

The growing success of deep neural network algorithms in solving challenging tasks resulted in a surge of interest from the safety-critical applications domain. As stated by recent works, one of the major issues of using such an algorithm in line with safety standards is the effects of label noise on the output [1–3].

Earlier object detection pipelines consisted of manually engineered feature extraction together with relatively simple classifiers [4,5]. These systems required a human to label the different objects for training, and the labeling was done

on the crop level. Although this approach had its challenges, such as mining negative examples, its behavior is still reasonably well understood due to relying on a straightforward method.

More recently, the success of convolutional neural networks (CNN) and deep learning [6] has transformed the domain of object detection. These approaches outperform traditional techniques by a large margin but are also more data-hungry at the same time [7–9]. The tedious task of manual labeling of enormous datasets means there will be faults in the process inevitably.

Popular large object detection datasets include MS COCO [10], PASCAL VOC [11], and OpenImages [12], containing millions of examples with quality annotations. The ground truth human annotations are gathered by crowdsourcing, and elaborate reward and evaluation schemes guarantee high quality for the annotations. However, apart from these large annotation campaigns, many players, companies, and research groups routinely collect smaller datasets within their application domains. In such cases, the quality of annotations is often compromised due to limited resources. Moreover, even standard benchmark sets are not error-free, and the influence of erroneous annotations on the system's safety requires further study.

The presence of noise in the training dataset can have a severe impact on the system's performance. For example, in the video surveillance system, a good detector would retain the same confidence, box coordinates, and class label over time. On the other hand, a bad one will be flickering, where the confidence fluctuates, the coordinates change, and even the class is mislabeled from frame to frame.

Figure 1 shows the four most common annotation error categories found in object detection datasets. These categories are (a) missing annotations (false negatives), (b) extra annotations (false positives), (c) inaccurate bounding boxes (which would result in low intersection over union (IoU)), and (d) incorrect class labels. In our experience, the most common error type is the first one, where the human annotator misses some target objects due to occlusions, small size, a large number of objects, or simply unclear annotation instructions. The second most frequent annotation error type is inaccurate bounding boxes, a very natural error for a human, as it takes more time and effort to pay attention to detail in every case. The two other types in Figure 1, completely incorrect annotations and wrong labels, are probably easier for humans to avoid.

The loss functions being a significant differentiator in modern single-stage detection pipelines and current challenges for annotation quality, inspired us to study the effect of label noise in object detection with two popular loss functions. Notably, in this paper, we focus on examining how *cross-entropy loss* (CE) and *focal loss functions* (FL) handle noise in the form of missing labels. We focus on these losses since the focal loss is commonly used but may suffer from missing annotations because it puts higher weight on complex samples (hard negatives and hard positives). Missing bounding boxes in the annotation appear as hard negatives from the training point of view, and we wish to study their influence on the resulting accuracy. The main contributions of this paper are:

- We characterize different types of noise present in object detection datasets.
- We provide empirical observations on training single-stage object detectors with different loss functions and different hyperparameter settings.
- We suggest possible measures to boost the robustness of the object detector with minimal changes in the network.

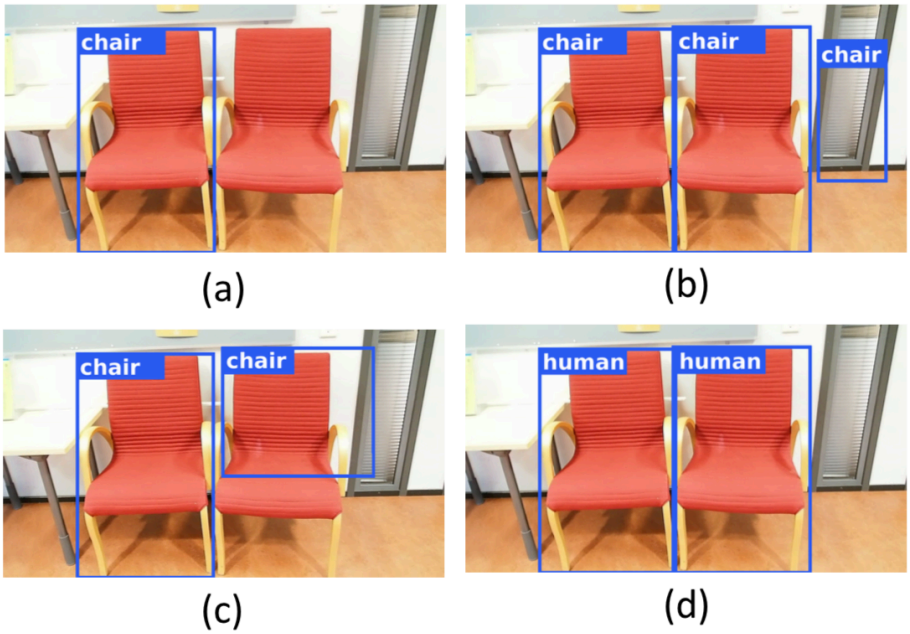


Fig. 1. Common types of label noise in object detection. (a) Missing label, the other chair is not labeled. (b) Incorrect annotation. (c) Inaccurately drawn box, resulting in low IoU. (d) Wrong classification label, humans instead of chairs. Image from the Indoor dataset [13].

The remainder of this paper is structured as follows. Section 2 briefly summarizes the related works followed by the review of object detection loss functions in Section 3. In Section 4, we experimented with multiple scenarios on our hypothesis and analyzed the obtained results. Finally, we conclude this paper with our findings and future direction in Section 5.

2 Related Works

Willers [1] and Wozniak [2] both provide a list of safety concerns or goals related to deep learning algorithms. In their works, label noise is mentioned as one of the primary faults that can affect safety. It is suggested to have a labeling guideline to

mitigate the effect of this fault. However, even with a guideline, manually labeling a large dataset is prone to noise, as discussed before. Thus, a proper approach is required to deal with noisy datasets in deep learning systems. Zhang [14] reviews problems related to the dataset, such as label noise, by surveying over recent works. According to his work, using a robust loss function and reweighting samples can help mitigate this issue.

Our topic of label noise in object detection is closely related to the topic of label noise in image classification, which has been studied more: For image classification, Frenay and Verleysen [15] have proposed a taxonomy of different types of noise, studied their consequences, and reviewed multiple techniques to clean noise and have the algorithms be more noise-tolerant. Li *et al.* have proposed BundleNet exploiting sample correlations by creating bundles of samples class-by-class and treating them as independent inputs, which acts as a noise-robust regularization mechanism [16]. Lee *et al.* have proposed CleanNet to detect noise in the dataset and be used in tandem with a classifier network for better noise tolerance [17].

Noise in object detection is different from classification because an image can have any natural number of objects present, anywhere in the image. A label in object detection is a box with a position, a size, and a class, which adds more possibilities for noise. It is easier for a human annotator to identify that an object in a picture is indeed a banana than correctly labeling dozens of bananas in one image of a cafeteria. The tedious task of doing so might result in the human annotator skipping some labels. Skipping a label causes label noise in the form of a missing label. Moreover, the task is often ambiguous when dealing with objects in a real-world image. Partially occluded objects, reflective surfaces, distance to the camera, and overcrowded images become relevant consideration points when labeling for object detection. These problems make the human annotator's role more prominent because more mental decisions are required. It also means that there will be more variation in the annotations, as different humans make different decisions.

Su *et al.* [18] have studied the overall process of annotation for object detection in a crowd-sourced manner. They first divided the task into three different sub-tasks: (1.) draw a box, (2.) verify the quality of a drawn box, and (3.) verify a box coverage on a single image. Different people do all these sub-tasks via Amazon Mechanical Turk (AMT). They concluded that this method produces good quality annotations.

Russakovsky *et al.* [19] have studied the human-in-the-loop annotation process, where state-of-the-art object detection models are used to detect many of the objects in the image. Then humans are used for detecting all the objects that the models are unable to detect. This method is needed as no current object detection system is perfect, yet, and their goal is to have every object in the image annotated adequately. A properly annotated object should have a tightly fitted box and not an arbitrary margin of non-object space in the annotation. They conclude that their method of using humans and computer vision together was better than using either alone.

3 Object Detection Loss Functions

Single-shot detection (SSD) [8] uses both regression loss for bounding box regression and cross-entropy loss for classification. The cross-entropy loss for a sample with ground truth one-hot-encoded labels $\mathbf{y} = (y_1, y_2, \ldots, y_C)$ and predicted class confidences $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C)$ in a C-class classification problem is defined as

$$\mathrm{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c). \tag{1}$$

The focal loss was extended by Lin *et al.* [20] to handle difficult samples better. They show that this improvement can result in better accuracy compared to the cross-entropy loss. The focal loss was designed to emphasize hard positives. It is similar to cross-entropy loss but has a parameterized penalty factor $\gamma > 0$ weighing the influence of each sample based on its detection score. More specifically, the focal loss for the C-class classification with ground truth $\mathbf{y} = (y_1, y_2, \ldots, y_C)$ and predictions $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C)$ is defined as

$$\mathrm{FL}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} \alpha_c (1 - \hat{y}_c)^{\gamma} y_c \log(\hat{y}_c), \tag{2}$$

with the balancing factor α_c [20], which is equal to 0.75 for all $c \in \{1, \ldots, C\}$ in all our experiments.

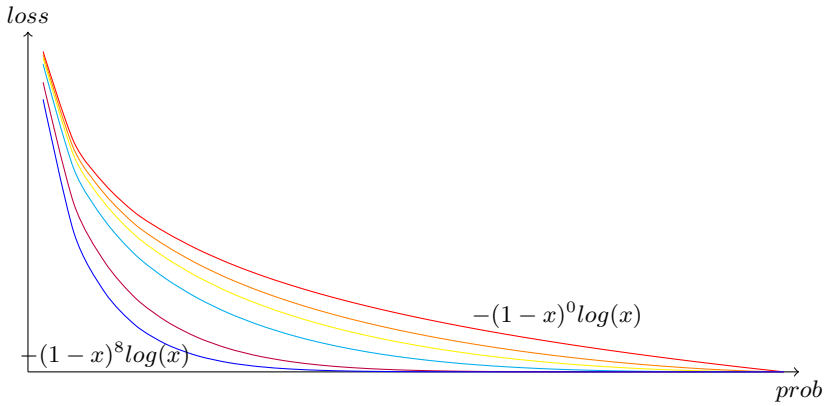


Fig. 2. Visualization of the focal loss function with different values for parameter $\gamma = 0, 1, 2, \ldots, 8$. The probability of being the ground truth is on the horizontal axis, and the loss is on the vertical axis. The higher the gamma value, the sharper the focus on harder cases. With gamma equaling zero, the focal loss is the same as cross-entropy loss.

In other words, the FL loss differs from the CE loss by the weight term $(1-\hat{y}_c)$, whose effect is to assign a higher weight for samples with low confidence (small

$\hat{y}_c$). γ affects the overall loss by lowering it; primarily well-classified samples with high confidence y_c for the most likely class c will have a negligible loss. At the same time, more attention is paid to learning the more complicated cases. Figure 2 demonstrates this scaling and aptly visualizes how the different γ parameters change the ferocity of the focus on more complicated cases. However, this loss weighting may have an adverse effect in the presence of label noise. The missing annotations are viewed as hard positives (non-annotated targets found by the model with a nonzero likelihood).

4 Experiments and Results

It was observed that sometimes in custom datasets, the focal loss seemed to produce results that were not as good as the research suggested. The intuition was formed that the weighting of complex cases, as performed by the focal loss function, would be more sensitive to label noise. The reason is that if a label is erroneous, to begin with, it is impossible to get right, so focusing on such a label leads the model astray and misuses the model capacity.

The experiments consider two questions: (1) how does label noise affect the two losses, and (2) how do models trained with different γ values tolerate label noise. For both experiments, we study the performance with three datasets: first with a small high-quality Indoor dataset, the second uses a large classical PASCAL VOC dataset, which does contain some annotation errors natively, and finally, with a single class FDDB dataset. Table 1 contains the characteristics of these datasets.

In all our experiments, the single-stage object detection (SSD) with MobileNet v1 [21] backbone network is fine-tuned from MS COCO pre-trained model for 100K training steps. We experimented only with the missing labels category. So, the training dataset has a percentage of randomly missing annotation boxes.

Table 1. Comparison of Indoor [13], PASCAL VOC 2012 [11] and FDDB [22] datasets based on source, size, quality of annotation, and usages.

	Indoor	**PASCAL VOC**	**FDDB**
Sample Source	Indoor scenes	Collected online	Faces in the Wild
Image Count	2213	17125	2845
Amount of Instances	4500	40000	5171
Number of Classes	7	20	1
Usage	Object detection	Multi-purpose	Face detection

4.1 Noise robustness of the two losses: CE vs. FL

In this experiment, we use six different noise levels: 0%, 10%, 20%, 30%, 40% or 50%, of missing labels. The dropping of the labels was done randomly, but both networks were using the same training datasets. Also, the noisy datasets are constructed incrementally, *i.e.*,, the 20% noise had all the labels of the 10% dataset dropped (+10% more), and so forth. The model with both the CE loss and the FL loss with hyperparameter $\gamma = 2$ (as proposed in the original paper [20]) is fine-tuned for 100K steps, and *mAP@.50IoU* (mean average precision with 0.5 IoU threshold) is used as a performance evaluation metric.

Indoor dataset— In the first set of experiments, we start training SSD using pre-trained weight from the MS COCO dataset, where some classes overlap between the datasets (chair, TV set, . . .), while others do not (fire extinguisher). The resulting accuracies are presented in Figure 3a; mAP@0.50 with the CE loss and the FL loss. Moreover, we show the *relative drop* in mAP with respect to noiseless labels in Figure 4. It seems that the accuracy resulting from the FL loss objective function outperforms the CE loss for 10% – 20% noise levels. The FL loss is more robust till the 30% noise level and maintains a higher mAP than the CE loss. However, with the higher amount of label noise (> 30%), FL loss accuracy plunged rapidly, falling behind the CE loss. For the extremely noisy (i.e., 50%) training dataset, accuracy from FL loss is 2% lower than that of CE loss.

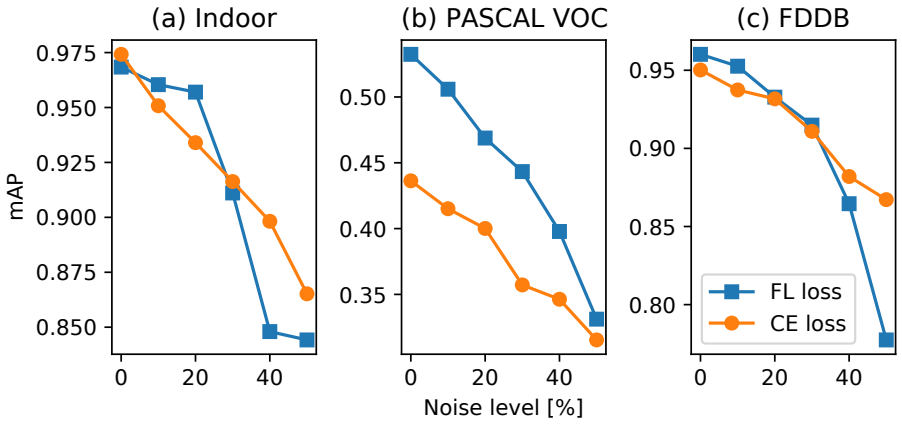


Fig. 3. Relationship between mAP (%) and different amount of noise levels on PASCAL VOC, and FDBB datasets.

PASCAL VOC dataset— Next, we studied the noise sensitivity on the PASCAL VOC [11] dataset. The network using the FL loss function performs better than the alternative, but the accuracy with FL loss decreases more when

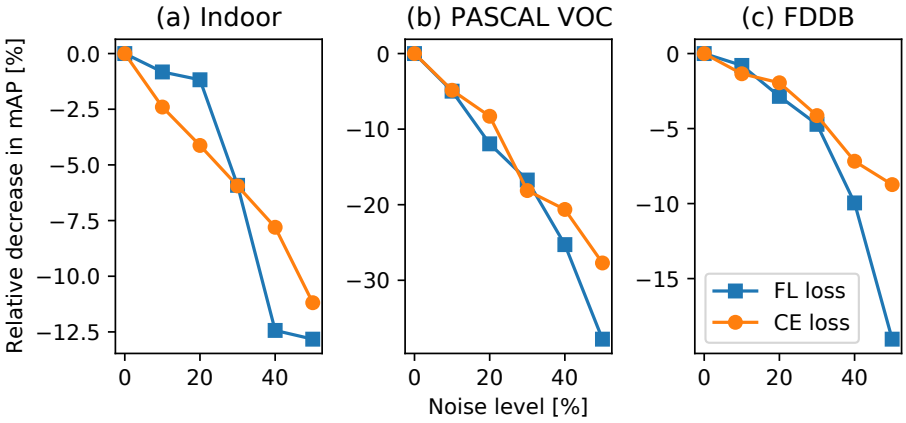


Fig. 4. Relative decrease in mAP (%) with respect to the noise levels in Indoor, PASCAL VOC, and FDDB datasets.

the noise level increases compared to CE loss. Without added noise, FL loss gives 10% higher mAP than CE loss. While the FL loss outperforms the CE loss in detection performance, it has a higher rate of mAP decrease than the CE loss. The difference in detection performance gets smaller by increasing label noise. The drop in mAP from no added noise to 50% label noise is 20.12% in with FL loss and 12.10% with CE loss.

FDDB dataset— Next, we studied the noise sensitivity on the high-quality moderate-sized single class dataset, Face Detection Data Set and Benchmark (FDDB)[22]. As shown in Figure 2c, the network using the FL loss function performs better till the 30% noise label. Adding more noise to the training dataset causes the accuracy to drop. The performance difference is smaller for lower noise levels and gets more significant for the noisy cases. The drop in AP from no added noise to 50% label noise is 18.28% in the FL loss case and 8.03% CE loss case.

Overall, the two losses seem to have similar behavior with these datasets. Compared to the FL loss, the CE loss is *more* robust to increased noise levels. However, with the VOC dataset, even though the FL loss suffers more for extreme cases, the overall performance remains higher than the CE loss at all points shown in Figure 2b.

We speculate that the Indoor and FDDB are relatively easy compared to VOC, containing fewer small (difficult) bounding boxes. Thus, as long as most bounding boxes are in place, the FL loss equally weights the true targets and the hard negatives produced by the missing labels. The more varied and challenging

nature of the PASCAL VOC dataset causes different noise tolerance behaviors than the smaller datasets.

4.2 Effect of the gamma parameter (γ)

In our second set of experiments, we compare the robustness of the FL loss for different values of the γ parameter. This time we only ran for three noise levels: 0%, 10%, and 50%. The gamma values tested were $\gamma = 1, 2, \ldots, 8$. All the other settings were kept the same as in the previous experiment.

Indoor dataset— The first experiment in this set uses the Indoor dataset; results on this dataset are presented in Figure 5a. In this dataset, the 10% noise detection performance is very close to the 0% noise. More interestingly, with extremely high label noise (50%), the gamma value has a significant impact. With $\gamma = 0$, the accuracy on the clean dataset (0% missing labels) is 18.52% more than the extremely noisy dataset (50% missing labels). With $\gamma = 8$, the clean dataset mAP is only 5.2% higher than the noisy dataset. The mAP curve indicates that a higher γ value does not affect the clean dataset while it boosts the performance in the presence of label noise.

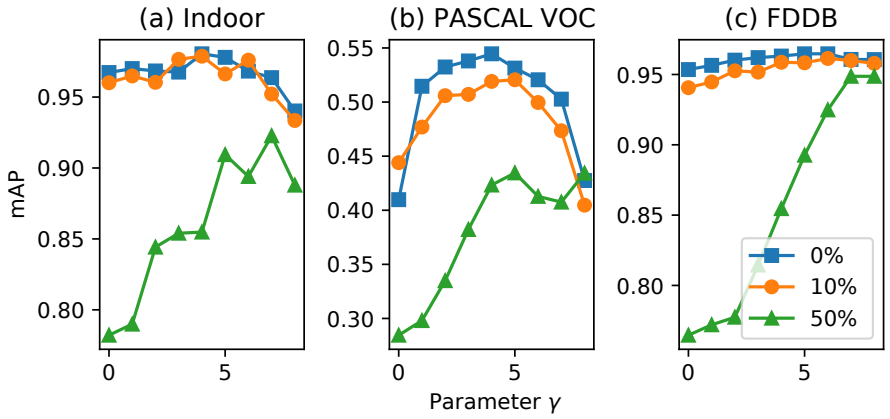


Fig. 5. Results on Indoor, PASCAL VOC and FDDB datasets with different gamma values on 0%, 10% and 50% noise levels.

PASCAL VOC dataset— Next, we experiment on PASCAL VOC with different γ values. The higher values of gamma can be used to offset the effect of missing labels partially. Like the previous experiment, γ values in the range 4 – 6 have better performance.

FDDB dataset— Experiments result on FDDB with different γ values is shown in Figure 5c. Results coincide with our previous experiments. With $\gamma = 0$, the difference in performance between clean and extremely noisy datasets is 18.90%. However, this difference gets smaller by increasing the γ value. With $\gamma = 8$, a clean dataset is only 2% more accurate than a heavily noised dataset.

Generally, with an extreme amount of label noise, increasing the γ value improves the detection results. Still, the exact γ value and the detection performance are dependent on the dataset. This could indicate that maybe the sharp concentration introduced by the higher γ values can offset the missing labels in relatively easy datasets. Experiments on these datasets suggest that the robustness to label noise increases for larger γ values. In these cases, the model essentially learns from the complex samples only (annotated targets detected with low confidence and non-annotated targets detected with high confidence).

This is illustrated in Figure 6, which shows the FL loss curves for both negative and positive examples. Due to the large γ value, the intermediate values ($\hat{y} \in [0.3, 0.7]$) behave as a *don't care* region, and the model does not learn from samples falling into this zone. Since all learning is based on complex samples (similarly to the support vector machine), it will be enough to push all objects with annotations to the "don't care" region. On the other hand, all negative samples (including missing annotations) can safely reside in this zone, and the model essentially learns to ignore those.

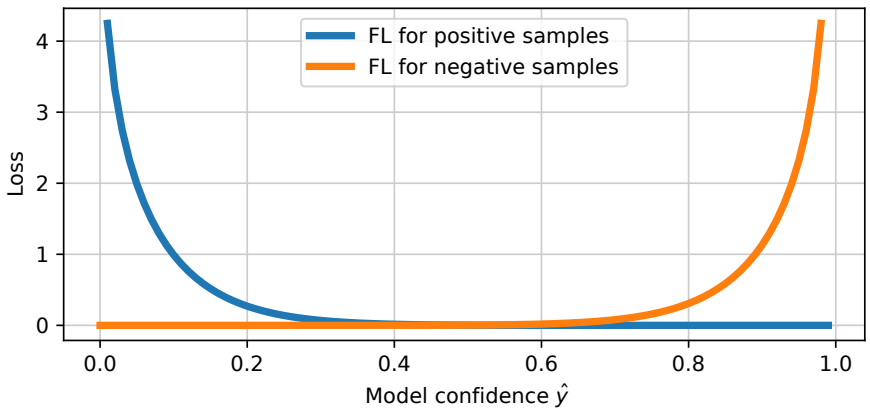


Fig. 6. Focal loss with $\gamma = 8$ for negative and positive samples with respect to model prediction confidence.

5 Conclusion

In this work, we characterized different types of label noise present in object detection datasets and explored the sensitivity of loss functions to them. With label noise being a crucial factor in ensuring the safety of the machine learning algorithm, we made sure to include experiments with large-scale real-world datasets. More specifically, we experimented on three datasets with varying amounts of label noise with cross-entropy and focal loss. Experiments suggest that focal loss suffers more with high amounts of noise, falling behind the cross-entropy loss. The second aspect studied is the effect of the hyperparameter γ on the sensitivity to label noise. It was discovered that larger values of γ improve the robustness to label noise such that extreme gamma values make the model indifferent to the noise level.

For future work, it would be beneficial to run more varied experiments to see how the label noise tolerance differs when training the network from scratch and its effect on system safety. Another point to consider would be running experiments with improved loss functions that are better suited for noisy datasets. It is also possible to quantify the risk associated with mislabeling by taking a statistical approach.

All relevant information, data, and codes are published open-access at `https://github.com/adhikaribishwo/label_noise_on_object_detection`.

Acknowledgment

This work was financially supported by Business Finland project *408/31/2018 MIDAS*.

References

1. Willers, O., Sudholt, S., Raafatnia, S., Abrecht, S.: Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks. Computer Safety, Reliability, and Security. SAFECOMP Workshops. 12235, 336-350 (2020).
2. Wozniak, E., Cârlan, C., Acar-Celik, E., Putzer, H.: A Safety Case Pattern for Systems with Machine Learning Components. Computer Safety, Reliability, and Security. SAFECOMP Workshops. 12235, 370-382 (2020).
3. Schwalbe, G., Knie, B., Sämann, T., Dobberphul, T., Gauerhof, L., Raafatnia, S., Rocco, V.: Structuring the Safety Argumentation for Deep Neural Network Based Perception in Automotive Applications. Computer Safety, Reliability, and Security. SAFECOMP Workshops. 12235, 383-394 (2020).
4. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). 1, 886-893, (2005).
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning. 20, 273-297 (1995).
6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. 521, 436-444 (2015).

7. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39, 1137-1149 (2017).
8. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., Berg, A.: SSD: Single Shot MultiBox Detector. European Conference on Computer Vision (ECCV). 9905, 21-37 (2016).
9. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 779-788 (2016).
10. Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.: Microsoft COCO: Common Objects in Context. European Conference on Computer Vision (ECCV). 8693, 740-755 (2014).
11. Everingham, M., Eslami, S., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The Pascal Visual Object Classes Challenge: A Retrospective. International Journal of Computer Vision. 111, 98-136 (2014).
12. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., Ferrari, V.: The Open Images Dataset V4. International Journal of Computer Vision. 128, 1956-1981 (2020).
13. Adhikari, B., Peltomaki, J., Puura, J., Huttunen, H.: Faster Bounding Box Annotation for Object Detection in Indoor Scenes. 7th European Workshop on Visual Information Processing (EUVIP). 1-6 (2018).
14. Zhang, X., Liu, C., Suen, C.: Towards Robust Pattern Recognition: A Review. Proceedings of the IEEE. 108, 894-922 (2020).
15. Frenay, B., Verleysen, M.: Classification in the Presence of Label Noise: A Survey. IEEE Transactions on Neural Networks and Learning Systems. 25, 845-869 (2014).
16. Li, C., Zhang, C., Ding, K., Li, G., Cheng, J., Lu, H.: BundleNet: Learning with Noisy Label via Sample Correlations. IEEE Access. 6, 2367-2377 (2018).
17. Lee, K., He, X., Zhang, L., Yang, L.: CleanNet: Transfer Learning for Scalable Image Classifier Training with Label Noise. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 5447-5456 (2018).
18. Su, H., Deng, J., Fei-Fei, L.: Crowdsourcing Annotations for Visual Object Detection. AAAI Human Computation Workshop. 40-46 (2012).
19. Russakovsky, O., Li, L., Fei-Fei, L.: Best of both worlds: Human-machine collaboration for object annotation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2121-2131 (2015).
20. Lin, T., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 42, 318-327 (2020).
21. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv (2017).
22. Jain, V., Learned-Miller, E.: FDDB: A Benchmark for Face Detection in Unconstrained Settings. Department of Computer Science, University of Massachusetts. UM-CS-2010-009 (2010).

PUBLICATION

V

Privacy-Aware Edge Computing System For People Tracking

J. Yrjänäinen, X. Ni, B. Adhikari and H. Huttunen

IEEE International Conference on Image Processing (ICIP), 2020, 2096–2100
DOI: 10.1109/ICIP40778.2020.9191260

PRIVACY-AWARE EDGE COMPUTING SYSTEM FOR PEOPLE TRACKING

Jukka Yrjänäinen, Xingyang Ni, Bishwo Adhikari, Heikki Huttunen

Faculty of Information Technology and Communication Sciences, Tampere University, Finland

ABSTRACT

This work studies the practical implementation of a distributed computer vision system for people tracking. A particular focus is on improved data privacy when compared to the traditional surveillance approaches. This is achieved by extracting a feature vector from each detected person by a neural network in real-time in the edge device and transmitting only the feature vector to the cloud, eliminating privacy-sensitive image data transmission and storage. The proposed solution is implemented in a network of Raspberry Pi single-board computers and Intel® Neural Compute Stick accelerators. The system is tested in an environment where multiple edge devices are sending data to the cloud server for further analysis. In this context, we consider the spectrum of design and implementation aspects of real-time execution of multiple neural networks in a capacity limited edge computing environment.

Index Terms— Computer Vision, Object Detection, Re-Identification, Neural Network, Edge Computing

1. INTRODUCTION

Computer vision is widely used for various surveillance tasks. In classical approaches, video streams are transmitted to a centralized location where data is stored for further analysis. Despite the precautions done to protect the data privacy in such systems, there is always a theoretical possibility that unauthorized persons could obtain access to the video data during the transmission or storage. Such concerns are also noticed by legislators, for example, in the European Union the General Data Protection Regulation (GDPR) has become enforceable at the beginning of 25 May 2018 and it sets strict demands for collecting personal information, including video surveillance data. This creates a demand for solutions that are able to benefit from recent advances in computer vision while taking into account personal privacy.

In this work, we propose a technical solution that serves various data analysis needs without compromising the privacy of the individuals. The solution is based on intelligent camera devices that process the data on edge, locating people with a neural network-based object detector and using another neural network for extracting abstract features of each individual. The detected objects' locations and corresponding features are sent to the server that collects the data feed from multiple cameras and stores them for further examination. The benefit of the proposed method is that features generated by the neural networks will make it possible to combine information from multiple cameras and time instances for analysis. However, it is not feasible to reconstruct the original picture from the feature vector, thus identifying the person as a named individual will be inherently extremely hard if one has access only to feature vector data.

The described approach assumes that there is enough computational power in the camera-equipped edge device so that we perform required processing with an adequate frame-rate in order to provide enough data for the analysis performed on the cloud. In this work, we study the practical implementation aspects of an edge device for such tasks. We analyze the computational cost of different deep neural network-based object detection and feature extraction algorithms on the edge device.

The rest of this paper is organized as follows: After a brief review of related work in the next section, we start with an overview of the used hardware and software solution in Section 3. The structures of used neural networks and their performance is studied in Section 4 and Section 5. Finally, we analyze the performance of the total system in Section 6 and conclude with concise summaries in Section 7.

2. RELATED WORK

Low-cost edge devices with neural network accelerators have been used for traditional surveillance applications in previous work. For example, Lage *et al.* study the use of object detection neural networks with Raspberry Pi and Neural Compute Stick [1]. Similarly, Dey *et al.* experiment with the implementation of pre-trained classification networks for the application of a robot vehicle navigation, and investigate the partition approaches of the processing between an edge device and a server [2]. These types of studies do not analyze the implementation of both object detection and feature generation in the same edge device. Furthermore, they do not concern the privacy issues involved with the transmission or storage of

This work was financially supported by Business Finland project 408/31/2018 MIDAS

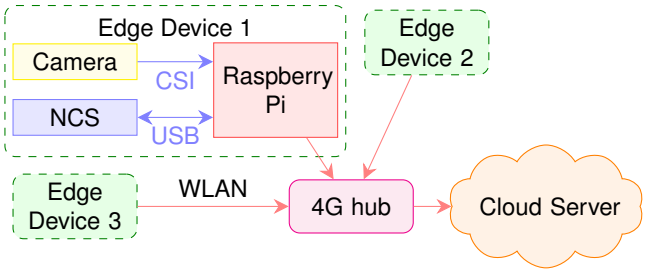


Fig. 1. System Overview. Key components of each edge device are camera module, Raspberry PI and Neural Compute Stick (NCS). Data are sent to the cloud server via WLAN and cellular connections.

sensitive image data.

Privacy challenges of machine learning based data analytics have been discussed in, e.g. [3] by Zhao *et al.*. Authors argue in favour of the approach in which the analysis is done on the edge device, reducing the exposure of the personal data to the cloud servers. While we agree with these statements, we add that when analysis requires composing information from multiple edge devices together, sending data to some centralized location cannot be avoided. In order to protect privacy, approaches, like described in our work, are required. Sophisticated Model Inversions Attack (MIA) techniques, as demonstrated by Fredrikson *et al.* in [4], are still a potential threat. However, in our approach as the feature calculation is done completely on the edge there is no need to expose the feature generation model to the cloud environment, making the practical implementation of any MIA strategy very hard.

3. SYSTEM DESCRIPTION

For the experiments, we did design and deploy the edge device fleet and the cloud server system as illustrated in Figure 1. Edge devices transfer data via WLAN and cellular modem to the cloud. On the server-side, data from multiple cameras are stored and analyzed according to the needs of the use case. For example, the feature vector data can be used for clustering all observations representing the same person. When analyzing timestamp and position information within a particular cluster, information about the duration of the visit and person's movements in observed locations can be retrieved.

In the physical set-up targeted in this work, devices need to be affordable, small, and easy to install at the location of use. Also, passive cooling is used since we want to avoid any disturbance caused by noise from the fan. Used hardware with the enclosure and typical installation position can be seen in Figure 2. The main components of the devices are Raspberry Pi 3 Model B+ single-board computers, equipped with 8 Mpix Raspberry Camera module V2 and Intel® Neural Compute Stick 2 (NCS) accelerator.

NCS is a fanless add-on accelerator that is connected to

Fig. 2. Hardware enclosure and typical installation position.

the host system via the USB interface. It is built around the Myriad™X Vision Processing system on chip (SoC) [5]. The chip contains 16 vision processing units (VPU) that are utilized for the acceleration of neural network processing. Programming happens with the OpenVINO [6] toolkit that offers SW tools to convert neural network models trained with popular frameworks to a format suitable for the NCS.

In our implementation, object detection is performed by neural network running on the NCS. The feature vector representing the detected person is computed by another neural network, either on ARM CPU of Raspberry or on the NCS. The Deep Neural Networks module of the OpenCV library [7] is utilized for fast inference on CPU.

4. OBJECT DETECTION

As the first step, the system needs to find bounding boxes associated with all persons in the camera view. For object detection, two commonly used neural network solutions are the Single Stage Detection (SSD) [8] and the Regions-CNN (RCNN) [9]. These two structures represent two widely used types of architectures, where the two-stage R-CNN is traditionally perceived as more accurate, especially with small targets. On the other hand, the single-stage SSD type networks are simpler, reach faster execution time, and still reach a reasonable accuracy when the targets are not exceptionally small. SSD can use different backbone networks for feature extraction, and the choice of backbone is an important design parameter balancing the trade-off between accuracy and speed.

When choosing which object detection architecture to use, we did initial speed tests comparing SSD and Faster-RCNN networks on the target architecture. Due to the limited computational budget, we select SSD framework for further study. We do analyze more in details the performance of 3 different SSD architecture variants: SSD with MobileNetV1 [10] and MobileNetV2 backbones and lighter version of SSD, i.e., SSDLite, with MobileNetV2 backbone[11].

As a starting point, we use pre-trained weights from models trained on MS COCO dataset [12], obtained from TensorFlow model zoo [13]. Fine-training of networks was done with person class images from OpenImages dataset [14]. We downloaded 76k(76561) images from the *train* and 1.8k (1874) images from the *validation* repository of OpenImages

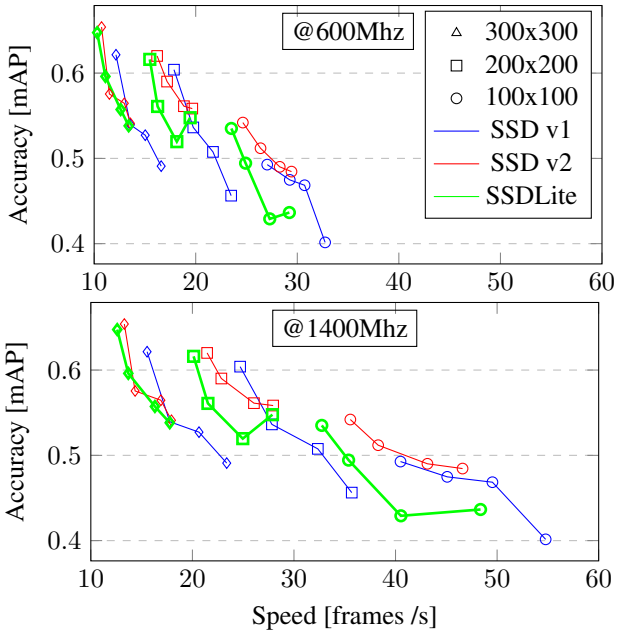


Fig. 3. Comparison of different SSD implementations running on NCS. Each solid line shows the effect of different depth multipliers (1.0, 0.75, 0.5, 0.25), while input size and network architecture do remain constant. The impact of CPU clock frequency (due image prepossessing on CPU and USB transfer speed) is seen in upper and lower graphs.

dataset person class, filtering occluded, truncated, depicted, and images taken from inside. The aim is to create a representative network capable of detecting persons in a wide range of different scenarios.

Multiple network configurations are analyzed. We test networks with input sizes of 300x300, 200x200 and 100x100. We also experiment with four values for depth multiplier (α = 1, 0.75, 0.5, 0.25), which impacts the number of channels used in layers of SSD. Figure 3 shows the performance of different architectures running on NCS. It is observed that the choice of input resolution and depth multiplier are giving a large set of trade-off options between accuracy and computational costs. Differences between network architectures are smaller, but it is noted that MobileNetV1 is faster at the expense of somewhat lower accuracy. We do consider two operating points for CPU: 1400 MHz "performance" and 600 MHz "power-save" mode. CPU frequency impacts the total performance of the detection via USB transfer speed and CPU computations used for image pre-processing, such as scaling.

5. FEATURE EXTRACTION FOR PEOPLE TRACKING

As we want to associate detected people across multiple cameras and moments time, we choose a *person re-identification*

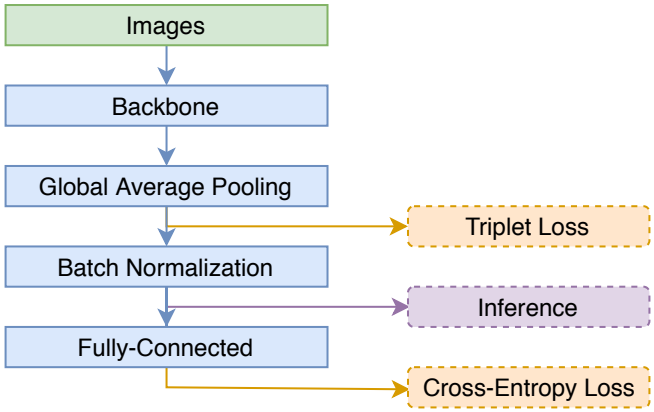


Fig. 4. The pipeline of the person re-identification algorithm in [15]. Two loss objectives are simultaneously optimized in the training procedure: triplet loss and cross-entropy loss. Subsequently, the output of the batch normalization layer is selected as the embedded features.

neural network as a means to provide features that are sent to the server. We adopt the person re-identification algorithm [15] trained on the Market-1501 [16] dataset.

Figure 4 illustrates the pipeline of the aforementioned method. A backbone model such as MobileNet [10] or ResNet50 [17] computes the feature maps of the input images. The backbone model is initialized with pre-trained weights on the ImageNet [18] dataset. Afterward, the global average pooling layer calculates the mean value of each channel. The output is processed by a batch normalization layer followed by a fully-connected layer that produces the probabilities of each identity. The triplet loss function [19] is applied to the output of the global average pooling layer, while the categorical cross-entropy loss function takes the output of the fully-connected layer. In the inference stage, the output of the batch normalization layer represents the embedded features of the images, and the cosine distance metric is employed to calculate the distance between the feature vectors of two images. Several strategies are implemented to prevent the overfitting issue, namely, random erasing data augmentation [20], label-smoothing regularization [21] and ℓ_2 regularization.

For our experiments, we choose a re-identification network with MobileNetV2 [11] backbone as its computational complexity is suitable for real-time implementation in the edge device. We study the effect of 4 different input sizes (160x160, 128x128, 96x96) and 4 different depth multiplier (1.4, 1.0, 0.75, 0.5) for the accuracy and processing speed. Results are shown in Figure 5 that illustrates the speed vs. accuracy of different network configurations in relation to the underlying processing system (CPU or NCS) and clock speed (1400Mhz or 600Mhz).

Implementation options are giving a wide range of trade-

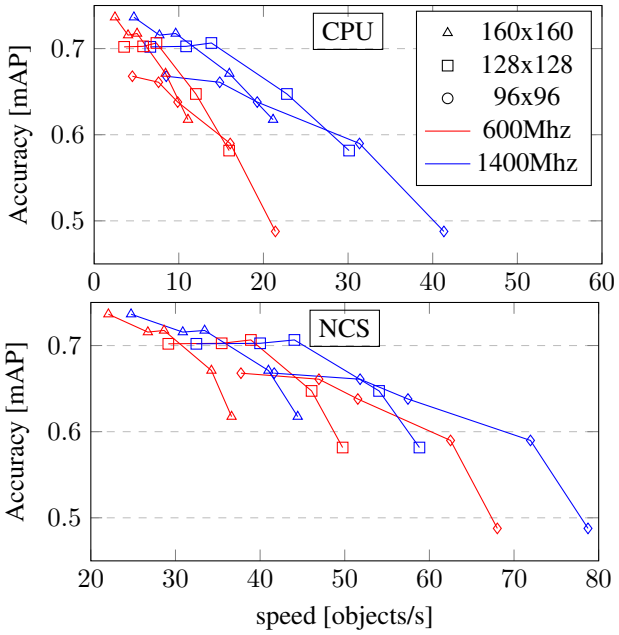


Fig. 5. Comparison between the implementations of the re-identification. Each solid line shows how different depth multipliers (1.4, 1.0, 0.75, 0.5) do impact to the performance, while resolution remains fixed. Upper graph shows performance on CPU and lower on NCS.

offs between speed and accuracy. The depth multiplier has a significant impact on both variables. With larger networks, the relative difference between CPU and NCS implementations is bigger compared to networks with low resolution and small depth multiplier, this indicates that if high accuracy is needed NCS is preferred, but in use case requiring less accuracy CPU is also a viable option.

6. SYSTEM PERFORMANCE

The performance of the system depends on the combined execution time of object detection and re-identification. As seen in previous chapters, choices with hyper-parameters are giving different trade-offs between accuracy and speed. The final choice is dictated by the use case. For example, when tracking people over a large number of cameras, high accuracy re-identification network is needed. On the other hand, in the set-up where there is a need to perform a single-camera tracking-by-detection scheme, a simpler feature generation network is sufficient, but a higher frame-rate for object detection is preferred. Likewise, for example, the object detection accuracy requirements are impacted by the assumed distance from person to the camera. At small distances, where object occupies a large area from the image, lower input resolution might be sufficient.

In our implementation, the total computing load is divided

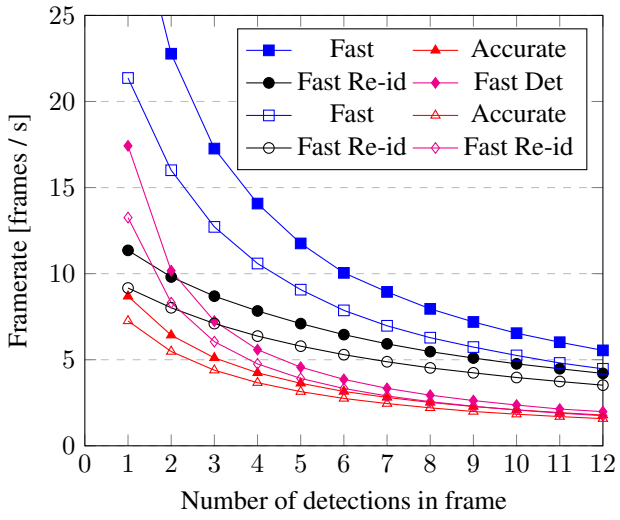


Fig. 6. Examples of the total system performance with different implementations of object detection and re-identification networks. All networks are running on NCS. Solid marker indicates 1400Mhz CPU frequency and open marker 600Mhz.

between ARM CPU and NCS. While NCS can do only neural network acceleration, CPU needs also take care of part of the pre-processing of the images and serving them over the USB to NCS. In addition to that, it needs to run the application tasks such as communication with the cloud server. In a such set-up, we advise running all networks in a dedicated accelerator.

Figure 6 illustrates measured system performance in conditions in which all networks are running on NCS. The graph shows various configurations: *'Accurate'*, where the SSD and re-identification networks with the highest mAP value are running together, *'Fast'*, in which fastest networks are executed and finally two trade-off configurations combining fast detector and with accurate re-identification and vice versa. The object count dominates the total processing time and restricts the available maximum frame-rate.

7. CONCLUSIONS

Our experiments demonstrate the feasibility of performing simultaneous object detection and feature generation using low-cost off-the-shelf hardware, enabling person tracking with improved privacy. We show that there is a large design spectrum for choosing optimal architecture for such tasks and invite further research efforts for privacy-aware computer vision. As technology advances rapidly, it is anticipated that even more processing power will be available for small edge devices. This makes the approach described in this work attractive for a wide range of real-world use cases.

8. REFERENCES

[1] E. S. Lage, R. L. Santos, S. M. T. Junior, and F. Andreotti, "Low-cost iot surveillance system using hardware-acceleration and convolutional neural networks," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, April 2019, pp. 931–936.

[2] S. Dey, J. Mondal, and A. Mukherjee, "Offloaded execution of deep learning inference at edge: Challenges and insights," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2019, pp. 855–861.

[3] Jianxin Zhao, Richard Mortier, Jon Crowcroft, and Liang Wang, "Privacy-preserving machine learning based data analytics on edge devices," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, New York, NY, USA, 2018, AIES '18, pp. 341–346, ACM.

[4] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2015, CCS '15, pp. 1322–1333, ACM.

[5] Intel, "Intel movidius myriad x vpu," `https://www.movidius.com/myriadX`, [Online; accessed 21-October-2019].

[6] Intel, "Intel distribution of openvino toolkit," `https://software.intel.com/en-us/openvino-toolkit`, [Online; accessed 21-October-2019].

[7] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Cham, 2016, pp. 21–37, Springer International Publishing.

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[11] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár, "Microsoft coco: Common objects in context," 2014.

[13] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," 2016.

[14] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv:1811.00982*, 2018.

[15] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang, "Bag of tricks and a strong baseline for deep person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, p. 0.

[16] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1116–1124.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[19] Alexander Hermans, Lucas Beyer, and Bastian Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

PUBLICATION VI

Towards a Real-time System for Facial Analysis

B. Adhikari, X. Ni, E. Rahtu and H. Huttunen

IEEE International Workshop on Multimedia Signal Processing (MMSP), 2021
DOI: 10.1109/MMSP53017.2021.9733663

Towards a Real-Time Facial Analysis System

Bishwo Adhikari[1] Xingyang Ni[1] Esa Rahtu[1] Heikki Huttunen[2]
[1]Tampere University, Finland [2]Visy Oy, Finland
{bishwo.adhikari, xingyang.ni, esa.rahtu}@tuni.fi heikki.huttunen@visy.fi

***Abstract*—Facial analysis is an active research area in computer vision, with many practical applications. Most of the existing studies focus on addressing one specific task and maximizing its performance. For a complete facial analysis system, one needs to solve these tasks efficiently to ensure a smooth experience. In this work, we present a system-level design of a real-time facial analysis system. With a collection of deep neural networks for object detection, classification, and regression, the system recognizes age, gender, facial expression, and facial similarity for each person that appears in the camera view. We investigate the parallelization and interplay of individual tasks. Results on common off-the-shelf architecture show that the system's accuracy is comparable to the state-of-the-art methods, and the recognition speed satisfies real-time requirements. Moreover, we propose a multitask network for jointly predicting the first three attributes, i.e., age, gender, and facial expression. Source code and trained models are available at https://github.com/mahehu/TUT-live-age-estimator.**

***Index Terms*—face detection, face recognition, facial similarity, real-time system**

I. Introduction

Human facial analysis is one of the widely studied topic in computer vision that includes face verification [1], [2], head pose estimation [3], [4], facial expression recognition [5], [6] and age estimation [7], to name a few sub-domains. While computer programs have traditionally been unable to analyze facial images, humans are very good at spotting even the smallest differences. With the surge of deep learning techniques, algorithms have surpassed human accuracy in most of the above tasks. More recently, deep learning has opened new avenues for applications of real-time facial analysis, such as facial identification [1], [2] and security surveillance [8].

In the field of facial image analysis, the majority of works focus on improving the accuracy of a specific task. Less attention is paid to investigate the computational complexity [9]–[11], in particular at the system level, where the architect needs to pay attention to the functionality of the entire system as well as that of the individual components; simultaneously optimizing for prediction accuracy, inference speed, memory footprint, parallelization as well as user experience (*i.e.*, the system should at least *appear* smooth although some components might operate at below real-time speed).

The straightforward approach would sequentially first detect all faces, then estimate their age, gender, facial expression, and facial similarity; show the result on the screen and start over with the detection. However, the refresh rate on screen would be dictated by the sum of execution times of individual components. On the other hand, users are less sensitive to a slow refresh rate of age estimates than the slow refresh rate of the display itself. Therefore, the system has to prioritize the tasks differently while maximizing the performance and minimizing idle times.

Fig. 1. Our real-time facial recognition system in action. It detects human faces on a frame captured by a webcam, recognizes age, gender, and emotion in real-time. Additionally, it shows the most similar appearing face obtained from the similarity search network.

Our system consists of a screen, a camera, and a computer, and it estimates the age, gender, and facial expression of all faces seen by the camera. In addition to these functions, the most similar-looking face from a database of celebrity faces is shown next to the detected face. Apart from serving as an illustrative example of modern human-level machine learning for the general public, the system also highlights several common aspects in real-time machine learning systems. The subtasks needed to achieve these recognition results represent a wide variety of tasks, including (a) face detection, (b) age estimation, (c) gender prediction, (d) facial expression prediction, and (e) image retrieval. Moreover, all these tasks should operate in unison, such that each task will receive enough resources from a limited pool.

Overall, we make the following contributions:

- We present a detailed system-level architecture for estimating several attributes from facial images.
- We show the real-time performance of each component of the proposed architecture and its smooth functionality even on a moderate-resourced computing platform.
- We release source code and trained models, with detailed instructions for deployment.

The structure of the rest of the paper is as follows. In Section II we describe the system level multi-threaded architecture for real-time processing. This is followed by a detailed description of individual components of the system in Section III. Next, we report the experimental setups

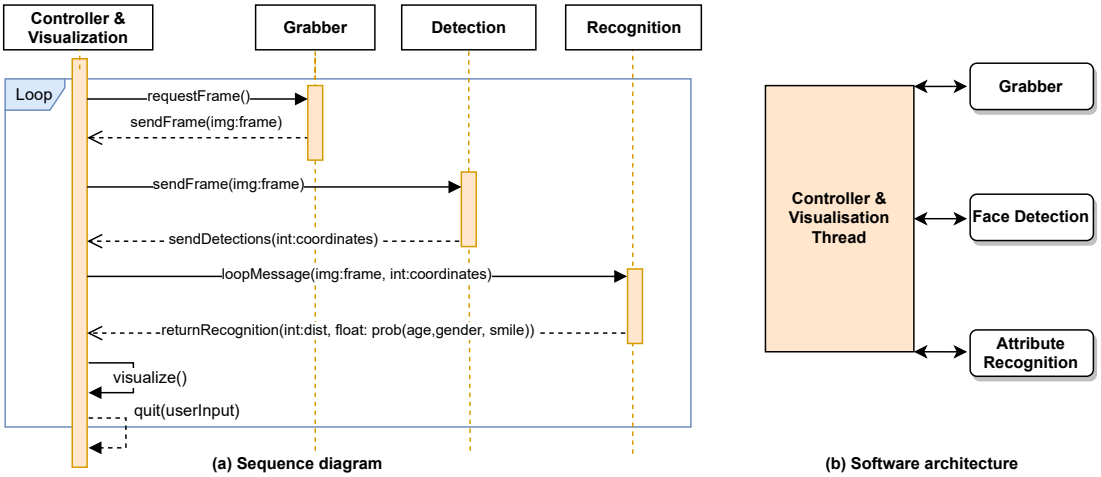


Fig. 2. Sequence diagram of the proposed real-time facial analysis system in (a) and software architecture of our system in (b).

together with datasets and performance measurement metrics in Section IV. We present experimental results of each recognition component in Section V and finally, we discuss the benefits of demonstrating the potential of modern machine learning to both the general public and experts in the field.

II. System Level Functionality

The challenge in real-time operation is that there are multiple components in the system, and each uses a different amount of execution time. The system should be designed such that the operation appears smooth, which means that the most visible tasks should be given higher priority in scheduling.

The implementation is multi-threaded, as illustrated in Fig. 2. Each thread operates asynchronously, with *recognition threads* polling for new frames to process whenever they are idle. The system is controlled by the *controller & visualization thread*, which receives new frames from the camera via the dedicated *grabber thread*. The controller thread also stores the frames in a buffer with each frame associated with flags, whether they have been processed by each of the threads. Finally, it visualizes by showing the live video as well as overlay the most recent recognition results to the user in real-time. The asynchronous threading structure also allows execution on dedicated platforms (*e.g.*, detection running on the CPU and recognition on the GPU). Also, it enables straightforward process prioritization by launching multiple recognition threads for the same task.

A. Frame Capture

The recognition process starts from the *grabber thread*, which is connected to a camera. The thread receives video frames from the camera for feeding them into a memory buffer located inside the controller thread. At grab time, each frame is wrapped inside a class object, which holds the necessary metadata: a time-stamp and flags indicating whether each of the processing stages (face detection, attributes recognition, and similarity search) has been applied on the frame.

B. Face Detection

The first processing step in the pipeline is to find all faces in the input frame. The detection is executed in a dedicated thread, which operates asynchronously, continuously requesting new non-processed frames from the controller thread. The detection algorithm is discussed in detail in Section III-A. Finally, the coordinates of the bounding boxes of all found faces are sent to the controller thread. The controller thread stores the locations and matches each new face with all face objects from the previous frames using straightforward centroid tracking. Tracking allows the system to temporally average the estimates (age, gender, and smile) for each face over a number of recent frames to improve the resulting accuracy.

C. Facial Attributes Recognition

The recognition thread is responsible for assessing the age, gender, facial expression, and facial similarity of each face crop found from the image. Like the detection thread, the recognition thread also operates in an asynchronous mode, requesting new non-processed (but face-detected) frames from the controller thread. When a new frame is received, the thread first aligns the face with a face template. After alignment, we pass each aligned face to separate networks: age, gender, and expression recognizer or a multitask and a similarity search.

Typically, the networks executed on the face crops are slower than the detection network. On the other hand, the amount of time grows linearly with the number of detected faces in the scene. Therefore, in order for the system to appear fast and responsive, these tasks should run in the background and only refresh when each task finishes. More specifically, we refresh the camera view and face detection in real-time but update the recognition results at less than the real-time rate. Moreover, the recognition thread prioritizes the facial expression task over others because age, gender, and facial similarity can be assumed to be constant, while users expect a quick response to their expressions.

The system is implemented using the TensorFlow and OpenCV libraries. The proposed facial analysis architecture can run on various hardware configurations, exploiting either CPU or GPU hardware. As shown in Section V, common desktop hardware reaches real-time speed both on CPU and GPU. However, if the camera resolution, detector type, or input resolution are changed, then a GPU can be used instead.

III. System Components

A. Face Detection

Face detection is the first step for facial recognition systems, where the location of the face is extracted from the given image. We design a neural network based face detector trained using benchmark face datasets. The detectors are not initialized from scratch but fine-tuned from existing pre-trained weights. We experimented with several models from two neural network based detection model categories: single-stage and two-stage detection networks. The single-stage Single Shot Detector(SSD) [12] requires only a single pass through the network with the image as the input and target bounding boxes with respective confidences as the outputs. The two-stage Regions Convolutional Neural Network (RCNN) [13] operates in two stages: a region proposal network proposes candidate object locations, followed by a classifier that classifies the proposals to target categories.

These two structures represent two widely used architectures, where the two-stage RCNN is traditionally perceived as more accurate, especially with small targets. On the other hand, the SSD type networks are simpler, reach faster execution time, and still achieve a reasonable accuracy when the targets are not exceptionally small. However, recent improvements [14], [15] in single-stage detectors have brought single-stage and two-stage architectures closer to each other, both in terms of accuracy and execution speed.

SSD model together with feature extractor networks such as MobileNetV1 [9] and MobileNetV2 [10] are popular for faster and light-weight object detection. MobileNetV1 introduced a parameter α called width multiplier to build a smaller and computationally efficient model. This width multiplier has the effect of reducing computational cost and the number of parameters quadratically by roughly α^2 times. MobileNetV2 introduced a mobile-friendly variant SSDLite that replaces regular convolutions with separable convolutions in the SSD prediction layers, reducing both parameter count and computational cost.

B. Alignment

We align the faces in two stages. The first stage locates a set of facial keypoints from the face crop: eyes, nose, and the corners of the mouth. In the second stage, we find an affine mapping between these five keypoint locations and the corresponding template of five keypoints. This improves accuracy since the recognizers always see the eyes, mouth, and other facial elements in fixed locations, and require less effort in understanding the context where facial features are located. This also enables the use of smaller networks, which compensate for the added computation due to the alignment procedure.

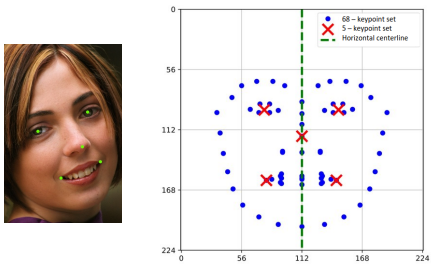


Fig. 3. An example of five-point facial keypoint on a cropped face region (left) and keypoint template (right). Symmetric keypoints are in blue dots, and the 5 referenced true keypoints are highlighted with orange color.

Keypoint Detection—The intention of aligning the faces to fixed coordinates is that this should improve the prediction accuracy. To this aim, we first find the keypoints for each face detected by the detector. We use five facial keypoints for normalizing the face location: eyes, nose, and the corners of the mouth, as illustrates in Fig. 3. Among the accurate and lightweight keypoint detection techniques, we consider regression forests of Kazemi *et al.* [16] and a convolutional neural network, where both receive the face crop as input and output the predicted x-y-coordinates of the five keypoints. We design the convolutional network according to the keypoint location branch of the O-Net [17]; consisting of four convolutional layers and two fully connected layers. The facial keypoint detector is trained from scratch on AFLW dataset [18].

Affine Mapping—The detected keypoints are aligned to a set of template keypoints. The template is obtained from the keypoints of a randomly selected sample face from the dataset. However, we normalize the template such that the keypoints are horizontally symmetric with respect to the centerline of the face. This is done in order to allow training set augmentation by adding horizontal flips of each training face. More specifically, we manually marked symmetric pairs of keypoints and averaged their vertical coordinates and distances from the horizontal center location as illustrated in Fig. 3. Finally, the resulting set of coordinates was scaled to fit the network input size of 224×224 pixels, leaving 10% margin at the bottom edge and 20% margin at the other edges.

Face Alignment—Instead of the simple approach of using the full affine transformation with least squares fit, we choose to use a more restricted *similarity transformation* allowing only rotation, scale, and translation, but not shearing. This is due to the possible distortion of the facial shape and the subsequent degradation of the estimation performances.

The similarity transformation $\boldsymbol{H}$ that maps 2D coordinate points $\mathbf{u} \in \mathbb{R}^2 \mapsto \mathbf{v} \in \mathbb{R}^2$ with translation $\mathbf{t} = (t_x, t_y)^T$, scaling $s \in \mathbb{R}_+$ and rotation matrix $\mathbf{R}$ with rotation angle $\theta \in [-\pi, \pi]$ is given by

$$\mathbf{v} = \boldsymbol{H}\mathbf{x} = \begin{bmatrix} s\boldsymbol{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{u} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u} \quad (1)$$

Estimation of the transformation parameters— $\mathbf{R}$, $\mathbf{t}$ and s— can be obtained from the vector cross product of point correspondences in homogeneous coordinates [19]. Given x-y-coordinates of detected keypoints $\mathbf{u}_i = (x_i, y_i, 1)^T$ and corresponding template locations $\mathbf{v}_i = (x'_i, y'_i, 1)^T$ for $i = 1, 2, \ldots, P$ (with at least $P = 2$ correspondences), the least squares solution for $\boldsymbol{H}$ can be obtained from the equation

$$\mathbf{v}_i \times \boldsymbol{H}\mathbf{u}_i = 0. \tag{2}$$

Substituting Eq. (1) into Eq. (2), the system is further simplified to [20]

$$\begin{bmatrix} -y_i & -x_i & 0 & 1 \\ x_i & -y_i & 1 & 0 \end{bmatrix} \begin{bmatrix} s\cos\theta \\ s\sin\theta \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} -y'_i \\ x'_i \end{bmatrix}, \tag{3}$$

which can be solved by the singular value decomposition [19]. Finally, we construct the similarity matrix $\boldsymbol{H}$ by inserting the four solved scalar unknowns into it.

C. Age Estimation

Age estimation is commonly treated as a regression problem. However, in our system, we treated this as a classification task as our system predicts ages among 101 classes. The network is initialized using ImageNet [21] pre-trained weights and fine-tuned in two stages: first with the large but noisy 500K IMDB-WIKI dataset [22] and then using the small but accurate CVPR2016 LAP challenge dataset [23].

D. Gender and Expression Recognition

The gender recognition network is trained from scratch in two stages: first with the 500K IMDB-WIKI dataset and then fine-tuned with the CVPR2016 LAP challenge dataset, same as in the age recognition step.

In our system, we focused only on smile recognition, a binary classification task, detecting smile and non-smile. The smile recognition network is initialized with ImageNet pre-trained weights and fine-tuned with Genki4k dataset [24].

E. Facial Similarity Search

In addition to the age, gender, and facial expression, the fourth analysis task integrated into the system is the facial similarity search. It is currently implemented for demonstrating *celebrity search*, *i.e.*, the program holds a database of celebrity faces and displays the one whose face has the most similar appearance to the person in front of the demo system. Alternatively, this functionality could be altered to keep track of users using a dynamic database (persons are added to the database every time they are seen) instead of a fixed database (a static collection of celebrities).

The facial similarity search is implemented in two stages: (1) the first stage computes a feature vector from the facial crop using a convolutional network, and (2) the second stage performs the nearest neighbor search among the database of precomputed feature vectors from celebrity faces. We use the FAISS implementation from Facebook [25], since it is widely adopted, provides an interface in Python, and satisfies the real-time speed requirement even with large databases.

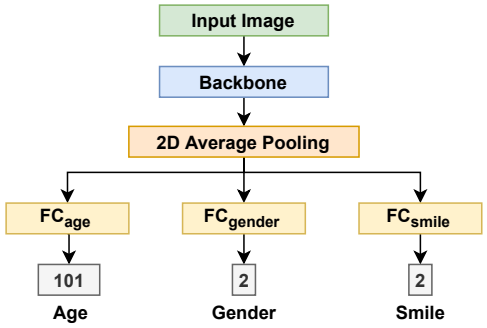


Fig. 4. The architecture of our multitask classification network. The network is able to classify age, gender, and smile attributes for a given image.

We adopt a person re-identification framework [26] to find the most similar face from a collection of celebrity faces. The backbone models are initialized with ImageNet pre-trained weights, and a global average pooling layer is appended to squeeze the spatial dimensions. Several data augmentation policies are applied to make the model more robust, including random flipping, cropping, and random erasing [27]. At the early stage of the training, the learning rate starts from a relatively low value and increases gradually. Additionally, the learning rate gets reduced to one-tenth once the performance on the validation split plateaus.

F. Multitask Network

We experimented with a single multitask network architecture shown in Fig. 4 for age, gender, and smile predictions. The multitask network utilizes a transfer learning approach; backbone network (ImageNet trained) weight is used to fine-tune on the age, gender, and smile training data. The backbone network can be any neural network for the classification task. The last layer of the network is removed, the global average pooling layer is added, and the output from the pooling layer is split into three branches. The fully connected layers, each of dimension 512 is added and SoftMax of different dimensions is applied for task-specific output branches.

Our multitask network inference time is almost identical to individual classification networks; hence this network is about three times faster than the individual networks for age, gender, and smiles recognition tasks, as reported in Table V.

IV. Experiments

A. Datasets

AFLW— The Annotated Facial Landmarks in the Wild (AFLW) [18] is a large-scale dataset of (25K) face images collected from Flickr. It has 21 landmarks annotations per face. We use this dataset to train 5 – keypoint set detection.

CelebA— The CelebFaces Attributes (CelebA) [28] is a large-scale face attributes dataset containing more than 200K celebrity images from 10,177 identities. It has 5 landmark locations and 40 binary attribute labels per facial image.

TABLE I
COMPARISON OF DIFFERENT DETECTION MODELS FOR FACE DETECTION WITH DIFFERENT INPUT SIZES.

Resolution	AP 0.5:0.95	AP @0.5	FPS TF-CPU	FPS TF-GPU	FPS OpenCV
Faster RCNN ResNet101					
300x300	**0.747**	**0.945**	1.84	7.62	1.09
240x180	0.707	0.914	1.93	8.18	1.20
200x200	0.693	0.907	**1.98**	**8.30**	**1.21**
SSD MobileNetV1 $\alpha = 1$					
300x300	**0.744**	**0.945**	32.52	87.29	39.36
240x180	0.684	0.868	**51.60**	105.46	70.09
200x200	0.683	0.839	49.96	**107.54**	**71.42**
SSD MobileNetV1 $\alpha = 0.25$					
300x300	**0.695**	**0.909**	60.50	148.77	140.95
240x180	0.647	0.895	**81.29**	156.62	239.59
200x200	0.650	0.887	78.21	**158.20**	**249.72**
SSDLITE MobileNetV2 $\alpha = 1$					
300x300	**0.764**	**0.952**	28.46	79.94	36.47
240x180	0.728	0.936	**41.18**	106.98	63.29
200x200	0.730	0.934	40.50	**109.47**	**64.79**
SSDLITE MobileNetV2 $\alpha = 0.25$					
300x300	**0.733**	**0.936**	43.09	118.29	70.58
240x180	0.704	0.925	60.43	129.25	125.55
200x200	0.679	0.913	**64.01**	**131.77**	**131.22**

ChaLearn LAP— We use the CVPR2016 competition variant, consisting of 7,591 facial images with human-annotated apparent ages and standard deviations taken in non–controlled environments with diverse backgrounds.

Genki-4k— The MPLab Genki-4k [24] contains 4,000 images with two class expressions (smile or non-smile) labeled by human and head-pose labels of the faces determined by automatic face detector.

B. Evaluation Metrics

AP—The Average Precision (AP) metric computes the average precision overall detection thresholds. The sensitivity of the detector can be adjusted using a detection threshold set by default at 0.5. As the sensitivity of detection may be adjusted at the inference process, we also average the class-wise AP's over all classes to produce the mean AP (mAP).

MAE—The Mean Absolute Error (MAE) metric computes the average error overall prediction. We used MAE to measure the error (in years) at the age prediction stage.

Accuracy—Accuracy is the fraction of correctly classified instances among the total number of instances.

CMC rank-k accuracy—Given a query sample, the accuracy is set to 1 if the top-k gallery samples contain samples that have the same identity as the query sample, and 0 otherwise. The CMC rank-k accuracy is obtained by averaging the results of all query samples.

V. RESULTS AND DISCUSSION

A. Face Detection

For face detection, we use faster RCNN, with ResNet101 backbone and variants of SSD, with MobileNet backbones with three different input sizes. The network inference speed is tested on Tensorflow CPU and GPU, and OpenCV environments, illustrated in Table I. The faster RCNN ResNet101 network in all experiments gives slightly higher AP than SSD models. However, the computation complexity of RCNN models is high, i.e., lower FPS compared to one-stage networks. Experiments show that with $\alpha = 0.25$, detection performance is about 3% less accurate while increasing inference speed about 1.5 times.

For all experimented networks, optimal detection performance is obtained by larger input size (*i.e.*, 300×300), while best FPS is obtained with smaller input size (*i.e.*, 200×200). With a smaller square input size, using OpenCV at inference always guarantees the best inference speed. If detection accuracy is not the top priority, using a small value of the α, smaller input size, and lighter model is suitable for faster and memory-efficient detection.

TABLE II
KEYPOINT DETECTION PERFORMANCE.

	Error rate(%)	FPS CPU
Dlib	2.89	17.49
CNN	**1.04**	**18.25**

TABLE III
PERFORMANCE IN FACIAL SIMILARITY USING THREE BACKBONES.

Network	mAP	rank-1	rank-5
MobileNet	0.782	0.940	0.970
VGG16	0.813	0.952	0.973
ResNet50	**0.822**	**0.953**	**0.973**

We measured the performance of two keypoint detection methods on the AFLW dataset. During the training, keypoint detection models were optimized based on the detected facial area with ground-truth keypoint labels.

Experiments on O-Net [17] based CNN alignment gives better alignment accuracy and inference speed as shown in Table II.

B. Facial Similarity

Our facial similarity system aims at finding the most similar face, and the rank-1 accuracy is a preferable evaluation metric. Table III shows the mAP, rank-1 accuracy, and rank-5 accuracy of facial similarity models trained with categorical cross-entropy loss on the aligned images from the CelebA dataset. The rank-1 accuracy of MobileNet reaches 94.0% which is slightly inferior to VGG16 and ResNet50, while using MobileNet is computationally lightweight.

C. Age, Gender and Expression Recognition

Table IV shows the accuracies of the different tasks included in our system. The speed test of each task on two different environments indicates that in the same environment, there is no significant difference between the network in terms of inference speed. However, the multitask network appears better considering the total inference time for three tasks.

The experimental results on our multitask network for age estimation, gender, and smile recognition with different backbone networks are reported in Table V. The best performing multitask network gives 5.35 years age MAE which is slightly higher than the best performing individual age network. Also, gender and smile accuracies obtained from the best performing multitask network are slightly less accurate than the individual networks. EfficientNet [11] networks give better age MAE and

TABLE IV
ACCURACIES AND INFERENCE SPEED AT DIFFERENT STAGES IN OUR SYSTEM. THE DEPTH MULTIPLIER $\alpha = 1.0$ IS USED IN ALL MOBILENETV1 NETWORKS.

Stage Network	Accuracy	FPS CPU	FPS 1050 TI	FPS 1080 TI
Age MobileNetV1	4.9 MAE	31.61	148.90	147.44
Gender MobileNetV1	88.3%	31.48	150.45	149.75
Smile MobileNetV1	87.2%	31.46	148.84	148.78
Multitask MobileNetV1	5.67 MAE 84.2% Gender 83.6% Smile	29.80	147.06	147.20
Multitask EfficientNetB0	5.35 MAE 87.5% Gender 86.0% Smile	25.61	143.35	144.24

TABLE V
PERFORMANCE COMPARISON OF THE MULTITASKING NETWORK WITH DIFFERENT BACKBONE NETWORKS.

Network	Age MAE	Gender Acc(%)	Smile Acc(%)	FPS CPU
VGG16	7.20	84.0	84.1	27.75
ResNet50	6.42	82.1	81.2	27.06
ResNet18	6.02	82.4	82.8	29.63
MobileNetV1	5.67	84.2	83.6	**29.80**
EfficientNet B0	5.35	87.5	86.0	25.61
EfficientNet B1	5.07	87.8	86.8	22.72
EfficientNet B7	**4.37**	**89.5**	**87.3**	14.63

recognition accuracies, but they are computationally expensive. As our computational budget is limited, and we cannot use a combination of many networks.

We can set up a system with a combination of a lighter detection model and a faster multitask network if the network accuracies are not the top priority. This way, real-time inference speed can be achieved on the CPU while slightly compromising each task's accuracy.

VI. CONCLUSION

We present a system-level design of a human facial analysis system with a multi-threaded architecture to reach real-time operation on resource-limited devices. We describe individual components of our system, integrating several standard machine learning components with an extensive set of experiments on each task. Users can switch specific task networks from the list of available options on the fly. The demo system has been presented several times in public locations. It has shown its value in illustrating the potential of modern machine learning in an easy-to-approach use case working on many levels. Moreover, this system can be used in the surveillance system by adding an alarm function that triggers once the detected face is matched with the suspect's faces on the query dataset. Additionally, the system can be used as a reference and baseline for related applications.

REFERENCES

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *CVPR*, 2014.

[2] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *ICCV*, 2009.

[3] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, 2009.

[4] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *CVPR*, 2011.

[5] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image and Vision Computing*, vol. 27, no. 6, pp. 803 – 816, 2009.

[6] P. Liu, S. Han, Z. Meng, and Y. Tong, "Facial expression recognition via a boosted deep belief network," in *CVPR*, 2014.

[7] Y. Dong, Y. Liu, and S. Lian, "Automatic age estimation based on deep learning algorithm," *Neurocomputing*, vol. 187, pp. 4–10, 2016.

[8] J. Yrjänäinen, X. Ni, B. Adhikari, and H. Huttunen, "Privacy-aware edge computing system for people tracking," in *International Conference on Image Processing (ICIP)*, 2020, pp. 2096–2100.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.

[11] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *(ICML)*, ser. Proceedings of Machine Learning Research. PMLR, 2019.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[14] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[15] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *ICCV*, 2017.

[16] V. Kazemi and S. Josephine, "One millisecond face alignment with an ensemble of regression trees," in *CVPR*, 2014.

[17] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, pp. 1499–1503, 2016.

[18] P. M. R. Martin Koestinger, Paul Wohlhart and H. Bischof, "Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization," in *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] J.-K. Kamarainen and P. Paalanen, "Experimental study on fast 2D homography estimation from a few point correspondences," Lappeenranta University of Technology, Tech. Rep. 111, 2009.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[22] R. Rothe, R. Timofte, and L. V. Gool, "Deep expectation of real and apparent age from a single image without facial landmarks," *International Journal of Computer Vision (IJCV)*, 2016.

[23] S. Escalera, M. T. Torres, B. Martínez, X. Baró, H. J. Escalante, I. Guyon, G. Tzimiropoulos, C. Corneanu, M. Oliu, M. A. Bagheri, and M. Valstar, "Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016," in *CVPRW*, 2016.

[24] S. D. MPLab, Uiversity of California, "The MPLab GENKI Database, GENKI-4K Subset."

[25] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[26] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang, "Bag of tricks and a strong baseline for deep person re-identification," in *CVPRW*, 2019.

[27] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[28] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.

PUBLICATION VII

Prediction of Future Paths of Mobile Objects Using Path Library

H. Leppäkoski, B. Adhikari, L. Raivio and R. Ritala

Open Engineering 11.1 (2021), 1048–1058
DOI: 10.1515/eng-2021-0103

Research Article

Helena Leppäkoski*, Bishwo Adhikari, Leevi Raivio and Risto Ritala

Prediction of future paths of mobile objects using path library

https://doi.org/10.1515/eng-2021-0103
received October 31, 2020; accepted May 19, 2021

Abstract: In situational awareness, the ability to make predictions about the near future situation in the area under surveillance is often as essential as being aware of the current situation. We introduce a privacy-preserving instance-based prediction method, where a path library is collected by learning earlier paths of mobile objects in the area of surveillance. The input to the prediction is the most recent coordinates of the objects in the scene. Based on similarity to short segments of currently tracked paths, a relative weight is associated with each path in the library. Future paths are predicted by computing the weighted average of the library paths. We demonstrate the operation of a situational awareness system where privacy-preserving data are extracted from an inexpensive computer vision which consists of a camera-equipped Raspberry PI-based edge device. The system runs a deep neural network-based object detection algorithm on the camera feed and stores the coordinates, object class labels, and timestamps of the detected objects. We used probabilistic reasoning based on joint probabilistic data association, Hungarian algorithm, and Kalman filter to infer which detections from different time instances came from the same object.

Keywords: path prediction, people tracking, probabilistic data association, instance-based learning, computer vision

*** Corresponding author: Helena Leppäkoski,** Tampere University, Faculty of Engineering and Natural Sciences, Tampere, Finland, e-mail: helena.leppakoski@tuni.fi
Bishwo Adhikari: Tampere University, Faculty of Information Technology and Communication Sciences, Tampere, Finland, e-mail: bishwo.adhikari@tuni.fi
Leevi Raivio: Tampere University, Faculty of Information Technology and Communication Sciences, Tampere, Finland, e-mail: leevi.raivio@tuni.fi
Risto Ritala: Tampere University, Faculty of Engineering and Natural Science, Tampere, Finland, e-mail: risto.ritala@tuni.fi

1 Introduction

The current developments in camera technologies and deep-learning-based signal processing have made computer vision systems cost-effective and feasible options for various surveillance tasks. In many applications, the ability to make predictions about the near future situation in the area under surveillance is as essential as being aware about current situation. For example, the security of the working environment of mobile machinery could be improved by a vision system that automatically detects objects in the area, predicts locations of the mobile objects, and determines how probably some parts of the area will be occupied in the near future. The ability to predict future locations allows, e.g., prediction of possible congestion in crowded areas or observation of atypical movement behaviors. For the operator of mobile machinery, the timely warnings provided with the help of these kinds of predictions could allow extra time to react when nearby mobile objects are about to enter to area of safety risk by approaching too close.

In this article, we introduce an instance-based prediction method, where a path library is collected by learning earlier paths of mobile objects in the area of surveillance, which preserves privacy of the tracked people. Many existing computer vision systems for surveillance rely on transmitting or storing video data to servers for further analysis. This compromises personal privacy of the people in the area. In our method, only position coordinates of the detected moving objects and the detection timestamps need to be extracted from the camera data. Any data that might identify the individuals are not stored or saved to the server.

We demonstrate the operation of a situational awareness system that uses only privacy-preserving data extracted from the computer vision system to track paths, learn the path library, and predict paths with the path library. Our system uses the limited set of data to answer the following questions: Are there people in the surveillance area? If there are, where are they and what kind of paths are they taking? Where will they be located, and which parts of the area will probably be occupied in the near future?

This article is organized as follows. Section 2 gives a brief summary of the reported work related to the topic and contents of this article. In Section 3, signal processing and algorithmic methods are described: processing of the camera-based observations to position coordinates, path tracking from time tagged coordinate samples, construction of the path library from the tracked paths, and using the recorded paths from the path library to predict the future paths of the moving objects in the camera scene. Section 4 describes the experimental setup to demonstrate the path prediction method, and finally in Sections 5–7 the results are reported and discussed and conclusions of the work are given.

2 Related work

The extension of Kalman filter (KF) techniques to prediction is a well-known approach [1]. Typically, the role of the KF in prediction is to propagate the system's state estimate and its covariance in time using the system's dynamic model that is represented in the form suitable for KF propagation equations. In some applications, the initial data sequence from the system is processed with the KF to obtain an accurate estimate of the initial state, from which the prediction can be started. For example, this approach is used in ref. [2] to predict the orbits of navigation satellites for 4 days ahead. Similarly, in our work we use the KF for initialization of both the library-based prediction and the KF prediction. However, the used models are different and in our work, other supporting methods, such as object detection and data association (DA), were required.

For pedestrian motion prediction from video stream, Schöller et al. [3] compared the prediction accuracy of simple constant velocity model without random perturbations to several state-of-the-art neural networks, e.g., long short-term memories (LSTMs), LSTM with state refinement (SR-LSTM), feed forward neural network, and social generative adversarial network (S-GAN). The conclusion of ref. [3] is that the simple model outperforms the neural networks in this task. Our work aims to solve the same problem, however, with different constraints (e.g., need for DA and randomly varying observation sampling) and targets to longer prediction lengths, and therefore we also use different methods. While in ref. [3] the number of time steps for initial observations and prediction lengths is 8 and 12, corresponding the time windows of 3.2 and 4.8 s, in our tests the length of the initial observation is 5–7 s (and samples) and the prediction lengths are 4–6, 9–11, and 19–21 s (and samples). We also enhance our prediction model by a collection of past paths stored into path library. The path library can be seen as an instance-based or memory-based learning method, a nonparametric method that uses the training instances themselves as a model [4].

Yrjanainen et al. [5] presented privacy-aware person tracking and counting on Raspberry Pi edge device together with the neural computing stick for smooth computation. They used object detection, person re-identification, tracking, and counting on the edge device and collected encrypted information over the network. However, while ref. [5] used a set of abstract features to identify the detected individuals, in this work, we did not use any individual identifying data at all. We deploy an object detection network on Raspberry Pi to collect only the object locations and the detection timestamps.

3 Methods

The path prediction using path library consists of several tasks. The tasks and the structure of their mutual connections are depicted in Figure 1.

First, the typical paths need to be learned, i.e., the path library needs to be collected. From camera images, the objects need to be detected and their locations estimated to obtain samples that include the coordinates of the detected objects and the timestamps of the detections. Without any data elements that identify the objects, it is not obvious which detections come from which objects. The missing link between the distinct samples is estimated by solving the DA problem: a new detection needs either to be associated with one of the existing tracked paths or to be found not to be a part of any of them, in which case it is treated as a start of a new path.

Once we know to which path the new detection is associated, we can track the path, i.e., use the position of the detection to update the path. We treat the path tracking task as a state estimation problem, where the motion model represents our assumptions on what kind of movements and motion changes are possible, i.e., what is the inertia of the object. We store the information on tracked paths to a database, which we call path library. A stored path is a sequence of positions and their estimated uncertainties.

To use the paths stored in the library for prediction, we need information on which paths might be relevant for the situation. To obtain that information, we need to run the path tracking to get an initial idea of what is happening in the camera scene. Once we have tracked

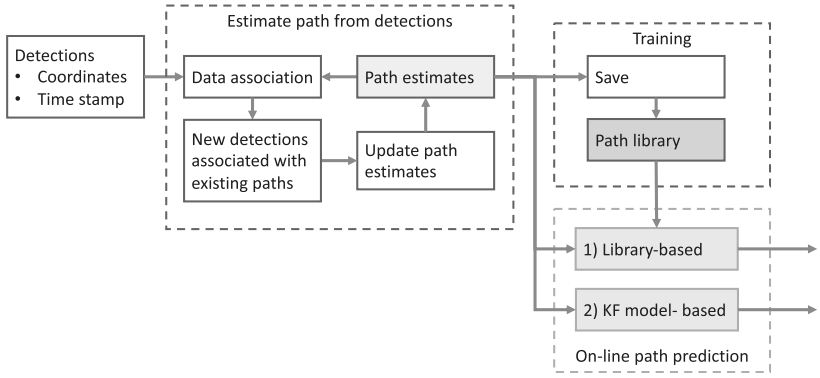


Figure 1: Main functional blocks of the path prediction system.

a new path for a couple of time steps, this short path segment can be compared with the contents of the path library. The library paths most similar to the new segment can be used to predict the future path. In the following subsections, these tasks are described in detail.

3.1 Object detection

The first step is to find the bounding boxes associated with the objects in the camera view. For object detection, convolutional neural network (CNN)-based methods have been proven to be effective with their state-of-the-art performances on public benchmark datasets [6–9]. These CNN-based object detection methods can be broadly divided into two groups: one-stage and two-stage. One-stage detectors are efficient and have straightforward architecture. In contrast, two-stage detectors have complicated architecture but perform better in terms of detection accuracy. Given an input image, the one-stage method directly outputs the object location and class without an intermediate proposal. The two-stage method explicitly generates region proposals followed by feature extraction, category classification, and finetuning of the location proposals. Single-Shot MultiBox Detector (SSD) [7], YOLO3 [6], and RetinaNet [8] are commonly used one-stage neural network solutions. Regional CNNs (RCNNs) [10] such as faster RCNN [9] and mask RCNN [11] are commonly used two-stage detectors.

For the detection, we use pretrained weights from the network trained on MSCOCO [12] and the fine-training of the network with our custom dataset, collected from our test environment mentioned in Section 4.2. Our Raspberry Pi system contains a single-stage object detection network, SSD MobileNetV2, that predicts object locations and a pre-defined class category for each detection. We then save object locations and class labels together with detection timestamps.

Since the data were captured with a stationary monocular camera, depth and – consequently – the distance of detected objects from the camera are unknown. Therefore, the three-dimensional locations of detected objects cannot be acquired directly. To estimate their location on the map, the objects are assumed to lie on a planar surface. A linear transformation is fitted by minimizing the Euclidean distance between known map points and estimates based on their respective positions in the image. Essentially, points in the image plane are transformed to a plane representing the ground surface and scaled to map coordinates. Objects are then projected to the map by applying the transformation to the center point between the two bottom corners of their bounding boxes, i.e., the part most likely to be connected to the ground. The camera is calibrated before the fitting procedure, and images are rectified before object detection and transformations. Although the transformation procedure is relatively simple, it works well in this study because the area in question is relatively small and flat.

3.2 Path estimation

An overview of the tasks involved in the path estimation and its data flow is shown in Figure 2. The basic tool in the path estimation is KF, which is described in many textbooks, e.g., in ref. [13,14]. The probabilistic reasoning for the DA uses the KF predictions, and the results of the DA are used in the KF state update. We presented the position coordinates of the detections and path points in *local-level-system* and its *east-north-up* (ENU) version [1]. As we assumed the objects to be located in a horizontal plane, we omitted the vertical coordinate. In the

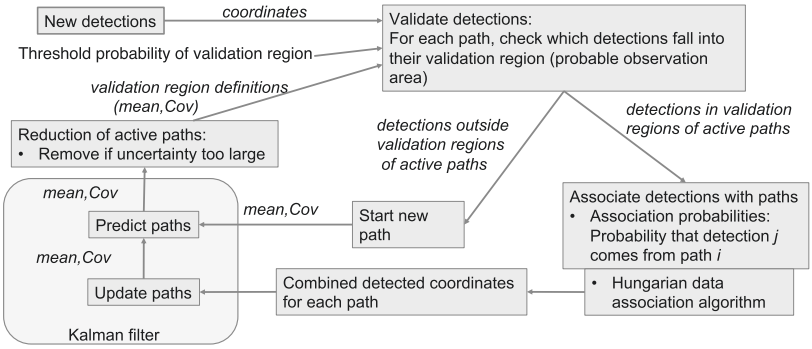


Figure 2: Subtasks of probabilistic reasoning for path estimation.

following, we denote the east and north coordinates as x and y, respectively.

3.2.1 System model

In the KF, we use constant velocity model as motion model to propagate the path estimates in time. The model consists of four states: position coordinates and velocities in two dimensions, written as $\mathbf{x}_k = [x(k), y(k), v_x(k), v_y(k)]^T$. The state is driven by zero-mean, Gaussian acceleration $\mathbf{w}(k)$. The discrete-time representation of the model is derived according to ref. [13]:

$$\mathbf{x}(k) = \mathrm{F}\mathbf{x}(k-1) + \mathbf{w}(k-1), \quad k = 1, \ldots$$
$$\mathrm{F} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where the variance of the driving noise $\mathbf{w}(k)$ is

$$E[\mathbf{w}(k)\mathbf{w}(i)^T] = \begin{cases} \mathrm{Q}, & i = k \\ 0, & i \neq k, \end{cases}$$
$$\mathrm{Q} = \begin{bmatrix} \frac{\Delta T^3}{3} & 0 & \frac{\Delta T^2}{2} & 0 \\ 0 & \frac{\Delta T^3}{3} & 0 & \frac{\Delta T^2}{2} \\ \frac{\Delta T^2}{2} & 0 & \Delta T & 0 \\ 0 & \frac{\Delta T^2}{2} & 0 & \Delta T \end{bmatrix} \sigma_w^2, \tag{2}$$

σ_w^2 is the variance of the driving noise, ΔT is the sampling time, and k is the index of the sampling instance. This model suits well for pedestrian motion, which forms the majority of the motion observed in our test environment. Other motion models that better describe the motion of objects with larger inertia and higher dynamics can be found, e.g., in publication [P1] of ref. [15].

The observation model is used for updating the path estimate. It describes the relationship between the true position of the object and the location of the detected object in the camera image projected onto map coordinates:

$$\mathbf{z}(k) = \mathrm{H}\mathbf{x}(k) + \mathbf{v}(k), \quad k = 1, \ldots$$
$$\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

where the variance of the measurement noise $\mathbf{v}(k)$ is

$$E[\mathbf{v}(k)\mathbf{v}(i)^T] = \begin{cases} \mathrm{R}, & i = k \\ 0, & i \neq k. \end{cases} \tag{4}$$

In our system, we used the following parameter values: $\Delta T = 1$ s and $\sigma_w^2 = 0.354^2$. As we had chosen the map coordinates so that the pointing angle of the camera was roughly to the direction of the negative x axis, and the projection of image coordinates to map coordinates is less accurate in the depth than width direction, we used higher variance for the x detection:

$$R = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}.$$

The choice of the values of covariances Q and R was based on our prior knowledge about the system, i.e., they were not systematically optimized or fitted to the data.

3.2.2 DA and path update

As the data did not include elements that link the latest detections to the earlier detections from the same object, we formed the link computationally as a solution of DA problem. For this task, we used joint probabilistic data

association (JPDA), a well-known method in, e.g., radar-based surveillance systems [16].

The first step of JPDA is the validation of the detections, where the task is to find which detected positions $\mathbf{z}_j(k)$, $j = 1,\ldots, n_z(k)$ are valid candidates to be associated with the existing paths, represented by their time-propagated estimates $\hat{\mathbf{x}}_i(k|k-\ell)$, $i = 1,\ldots, n_x(k)$. Here ℓ is the number of time steps the estimate is projected ahead after its last observation-based update, $n_z(k)$ and $n_x(k)$ are the numbers of detections and active paths at time index k, respectively.

The detection j is considered to be a valid association candidate to path i, if its observation likelihood with the path exceeds threshold P_G. As we assume that the errors of the detection positions and the path estimation errors are Gaussian, the validation region, i.e., the area where the likelihood condition holds, is an ellipse (illustrated in Figure 3). Its center is located in the position coordinates of the predicted path, $\hat{\mathbf{x}}_i(k|k-\ell)$, and its size and shape are defined by the covariance matrix of the error between the predicted observation $\hat{\mathbf{z}}_i(k) = H\hat{\mathbf{x}}_i(k|k-\ell)$ and the actual observation $\mathbf{z}_j$. Therefore, the innovation covariance $S_i(k|k-\ell) = HP_i(k|k-\ell)H^T + R$ is computed for each path i and innovation vector $\tilde{\mathbf{z}}_{i,j}(k) = \mathbf{z}_j(k) - \hat{\mathbf{z}}_i(k)$ is computed for all pairs of paths i and detections j.

For Gaussian vectors, the checking against a probability threshold with probability ellipses can be transformed to checking of Mahalanobis distances. The squared Mahalanobis distance of innovation is $d^2_{i,j}(k) = \tilde{\mathbf{z}}_{i,j}(k)^T(S_i(k|k-\ell))^{-1}\tilde{\mathbf{z}}_{i,j}(k)$. For a Gaussian 2D vector, the Mahalanobis distance follows $\chi^2(2)$ distribution. Therefore, instead of checking whether a point lies inside a probability ellipse, we can check that the Mahalanobis distance between the point and the center of the ellipse does not exceed the corresponding d^2 threshold. This threshold is obtained from $\chi^2(2)$ inverse cumulative distribution function: $d^2_{P_G} = F^{-1}_{\chi^2(2)}(P_G)$. In our experiments, we used $P_G = 0.99$ and the corresponding $d^2_{P_G} = 9.21$.

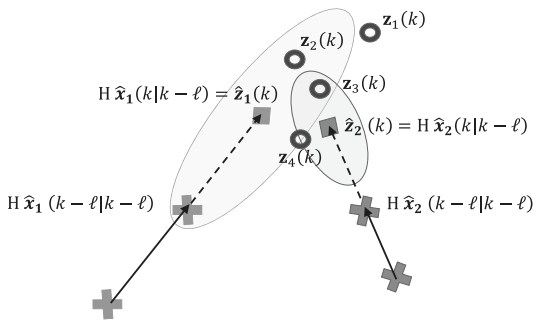


Figure 3: Example of validation regions. The four latest detections (the circles) and the elliptic validation regions of two paths. The crosses denote the path estimates updated by their associated detections and the predicted path positions based are shown with squares.

Based on the computed Mahalanobis distances, all detections $\mathbf{z}_{j'}$ for which $d^2_{i',j'}(k) \le d^2_{P_G}$ are considered valid association candidates for paths $\hat{x}_{i'}$. However, these are not necessarily correct associations, as there may be several detections in validation region or there may be overlapping validation regions when two or several paths share common candidates. It is also possible that more than one path have the shortest Mahalanobis distance to the same detection.

The next task is to find the association between the detections that are valid association candidates and the paths that have valid candidates in their validation regions. When assigning the detections to the paths, we allow at most one detection to be associated with at most one path, i.e., for each path and each detection, there is either one-to-one association or no association at all. We want to find the matching pairs so that the overall cost, i.e., the sum of the squared Mahalanobis distances between the associated detections and paths, is minimized. This type of linear assignment problem can be solved with Hungarian algorithm [17,18]. We used Matlab function `matchpairs` to solve the problem.

Once the association between the detections and path estimates is made, we run the KF observation update for the path estimates that have an associated detection. Often there are also detections that are not associated with a path, either because they were not valid candidates or because there were more valid association candidates than active path estimates. We treat these detections as initial observations of new paths. After all the detections are used to update either an existing or a new path, the path estimates are propagated in time using the motion model defined in (1) and (2), until new detections are available.

After the propagation steps, the uncertainties of the path estimates are checked. The product of the eigenvalues of the position covariance is computed and if it exceeds the uncertainty threshold, the path is removed from the set of active path estimates.

3.3 Path library

In the training phase of the prediction model, the paths removed from the set of active paths are stored into path library. For each path, the sequence of path position estimates and the corresponding covariance matrices were

stored. To prevent too short-lived path sequences from populating the library, we did not store sequences that were shorter than the threshold L_{pmin}. The sequence length was defined as the difference between the last and first observation update of the path estimate. For the sampling instances without observation update, the predicted estimate was stored, otherwise the updated estimate.

As we used only one camera, it is possible that an object may stay behind another object for several sampling instances, and has moved far from the edges of the camera view when it becomes visible for the camera for the first time. Therefore, we accept that a path can start everywhere in the camera view. Considering this condition, we added a grid representation into the path library to allow faster search of similar library paths in path prediction. We divided the area into 1 m × 1 m grid and to each grid cell, we stored the indices of the paths and the sequence indices of these paths that have coordinates located in the cell.

3.4 Prediction

The library-based prediction principle is illustrated in Figure 4. The input to library-based prediction is the most recent position coordinates of the objects in the scene. The paths are tracked from the detections with the KF model and JPDA techniques described in Section 3.2.2. When the lengths of such paths increase above a given threshold $\delta t_{\text{min}} = n_{\text{ip}}\Delta T$, the obtained path segment, which we call initial path, is compared to the contents of the path library.

Based on similarity with the initial path, a relative weight is associated with the paths of the library. However, to avoid giving any weight to the library paths very far away from the initial path and to save computational resources, we limit similarity comparisons using the grid representation of the library. We start scanning the library paths from the library grid cell where the initial path starts and then continue to the neighboring cells and further by increasing the distance to the starting cell, until the weights of n_{pmax} library paths are evaluated or all the cells within the Manhattan distance d_{cmax} to the starting cell have been checked.

The similarity between the paths is evaluated using the assumption of the Gaussian distribution of the path point coordinates and the common interpretation that the KF estimate and its covariance are the parameters of this distribution. In the following, an initial path and a library path are defined by sequences $\{\hat{\mu}_{\text{ip}}(k), \Sigma_{\text{ip}}(k)\}$ and $\{\hat{\mu}_{\text{lib}}(k), \Sigma_{\text{lib}}(k)\}$, respectively, where $k = 1, \ldots, n_{\text{ip}}$. Here $\hat{\mu}$ and Σ represent the position parts of the estimate and its covariance, i.e., $\hat{\mu} = \hat{\mathbf{x}}_{1:2}$ and $\Sigma = P_{1:2,1:2}$. First the Mahalanobis distances between the initial path and the library path are computed for $k = 1, \ldots, n_{\text{ip}}$:

$$d^2_{\text{ip,lib}}(k) = (\hat{\mu}_{\text{ip}}(k) - \hat{\mu}_{\text{lib}}(k))^T(\Sigma_{\text{ip}}(k) + \Sigma_{\text{lib}}(k))^{-1}(\hat{\mu}_{\text{ip}}(k) - \hat{\mu}_{\text{lib}}(k)).$$

From this, the corresponding values of $\chi^2(2)$ probability distribution function are computed:

$$p(k) = f_{\chi^2(2)}(d^2_{\text{ip,lib}}(k)).$$

The weight of the library path is obtained as

$$w_{\text{ip,lib}} = \exp \sum_{k=1}^{n_{\text{ip}}} \log(p(k)). \tag{5}$$

Using the weights (5), computed for all library paths that were found from the grid cells close to the start of the recently tracked initial path segment, the future path is predicted by computing the weighted average of the paths in the library and its uncertainty is expressed with its weighted covariance matrix. Quite commonly, the prediction is composed of several "branches," meaning that the distribution of predicted agent's location is multimodal. In this case, an appropriate descriptor of the uncertainty is the smoothed probability distribution produced by the Gaussian mixture that represents the branches.

If library paths that match closely enough with the initial path are not found, all the path weights are zeros and we cannot compute the library-based prediction. Then the algorithm has to revert to KF-based prediction as a fallback method.

In our experiments, we used the following parameter values: $\delta t_{\text{min}} = n_{\text{ip}} = 6$, $n_{\text{pmax}} = 50$, $d_{\text{cmax}} = 15$, and $L_{\text{pmin}} = n_{\text{ip}} + n_{\text{pred}}$, where n_{pred} is the number of time steps that the prediction is computed for.

4 Experiments

For real data demonstration, we collected camera-based object detections. We used part of the data to learn the paths to path library and another part to assess the quality of library-based prediction by comparing it to the KF-based prediction. In the following, we describe the hardware we used, the data we collected, and the test procedure for the prediction.

4.1 Hardware

We use affordable, portable, and easily available edge-device to collect experimental data and run the detec-

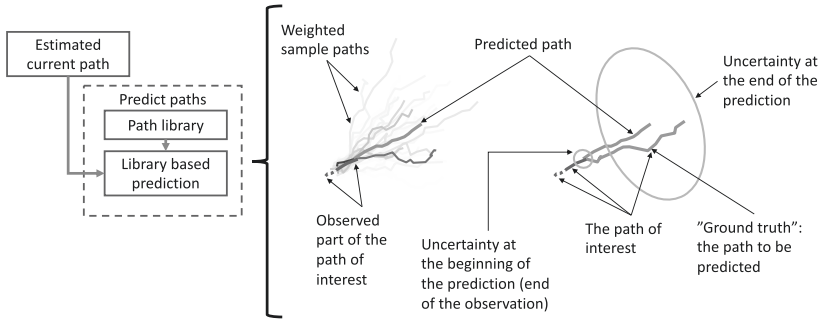


Figure 4: The predicted path is the weighted sum of the most similar library paths.

tion system. The system includes Raspberry Pi 3 Model B+ equipped with 8 Mpix Raspberry camera module V2. Cooling fan and heat sinks are attached to the Raspberry to prevent overheating and unexpected shutdown during continuous operation for days. The system shown in Figure 5 is placed on the fifth floor of the Tampere University Hervanta Campus building facing toward the open space next to the parking yard and capturing the view in Figure 6.

4.2 Data

The system described in Section 4.1 was used to collect full HD videos during daylight in summer 2018. About 800 frames were extracted from videos captured on different days and times. These image frames were fully labeled in six class categories: bus, car, cyclist, person, truck, and van, using the technique mentioned in ref. [19]. One annotation means two coordinate points and a classification, visualized as a box and a written class label. The aim was to create a representative dataset including various moving objects during daylight for the object detection network training. We emphasize that our system does not send any visual data to the server. Once the detection system is online, it sends the detected object's location coordinates and associated timestamps. However, we save some amount of visual data for system debugging and visualization purposes. Hence, the system preserves the privacy of the person being in the camera view.

In the dataset, "person" was the only category with plenty of instances spread over the scene. Therefore, we used only this category in the prediction tests. These detections appear in bursts and the intervals between the detection bursts vary randomly, the most common interval lengths being 2 or 3 s.

To create the path library, we used stored location and timestamp data collected on day 1 during 13 h. The data from the next day were used to compare the predictions produced by the path library and the traditional KF. The path tracking and prediction were implemented with Matlab.

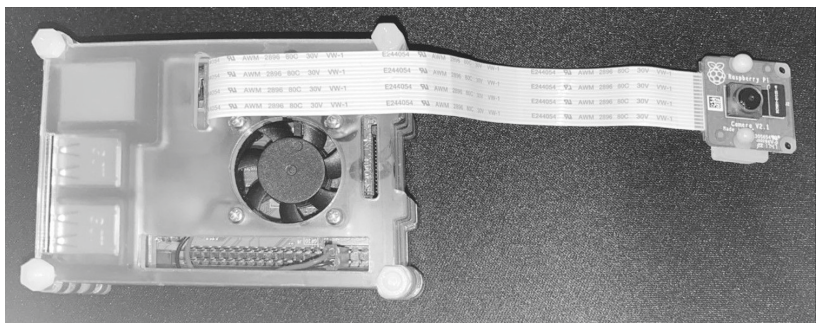


Figure 5: Hardware for the data collection and running detection network.

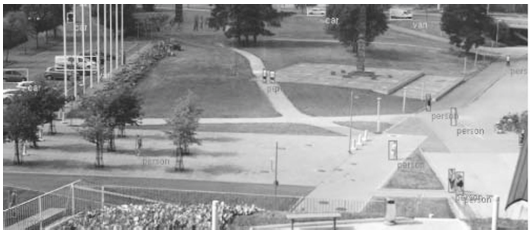


Figure 6: Camera view from the test setup running real time object detection on a Raspberry Pi platform.

4.3 Test procedure

In the prediction tests, we wanted to compare the predictions to the detections and to examine the performance of the method in different prediction lengths. We defined three interesting prediction lengths $t_{\text{pred}} = n_{\text{pred}}\Delta T$: for short-, middle- and long-term predictions, t_{pred} was 5 ± 1, 10 ± 1, and 20 ± 1, respectively.

The random variation of the sampling interval posed challenges to the testing, as to be able to assess a prediction made for a time instance, there should be detections available at the time. To tackle this problem, we used the timing shown in Figure 7. The value $\delta t_{\text{min}} = 6$ s was chosen to make sure that most often at least three detection instances will be included in the initial path. To allow comparison against detections, the predictions were computed and saved for three time instances around the targeted prediction lengths. Due to the variation in the detection sample intervals, the actual tracking interval δt varies as the tracking ends when the first detections are obtained such that $\delta t(m) \geq \delta t_{\text{min}}$. For the same reason, the age of the prediction that is compared with the actual detections also varies.

In the test, we stopped the path tracking and cleared the memory of the initial path after it was used to compute predictions. The future detections, possibly originating from the same actual paths were then used to start and track a new initial path, i.e., the detections from the same actual path were used to start predictions several times at different phases of the path build-up.

We compared the predictions produced by the path library to the predictions of KF. With KF we mean here the same KF combined with JPDA that produced the initial path for library-based method, but instead of using the initial path, the KF used its last estimated state and the motion model (1) to make prediction for the next n_{pred} time steps.

5 Results

To assess the capability of the library-based path prediction, it was compared to KF predictions. The criterion for the comparison was how probable the prediction method had considered the detection of objects in the location where they actually appeared. Looking at a location where an object became detected, the higher its predicted probability to be occupied was, the more successful we considered the prediction.

A captured moment of a tracking and prediction situation is shown in Figure 8. For illustration purposes, the build-up history of paths is shown even though for prediction, the path history was cleared after the predictions were computed. Some phenomena are marked with labels in the figure: (1) two adjacent paths; (2) a long, static sequence of detections; (3) a long path; (4) a short, static sequence of detections; (5) predictions indicate probable detections but the series of detections has ended; and (6) pink shade on the map denote areas where library paths have plenty of position occurrences. The expanding yellow circles represent the KF predictions and their uncertainty that increases with time. The blue circles represent the uncertainties of the library-based predictions, which do

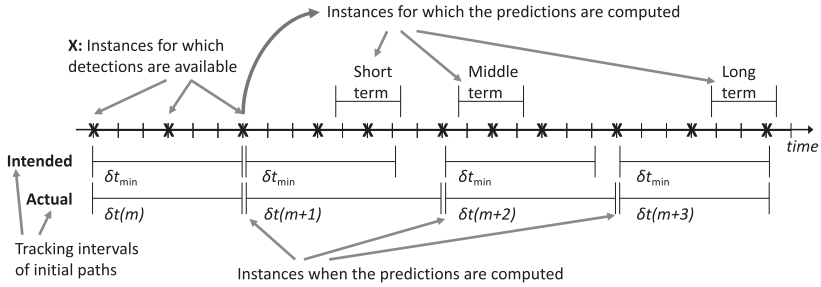


Figure 7: Timing diagram of the prediction test exemplifying the effect of the random variation of detection times.

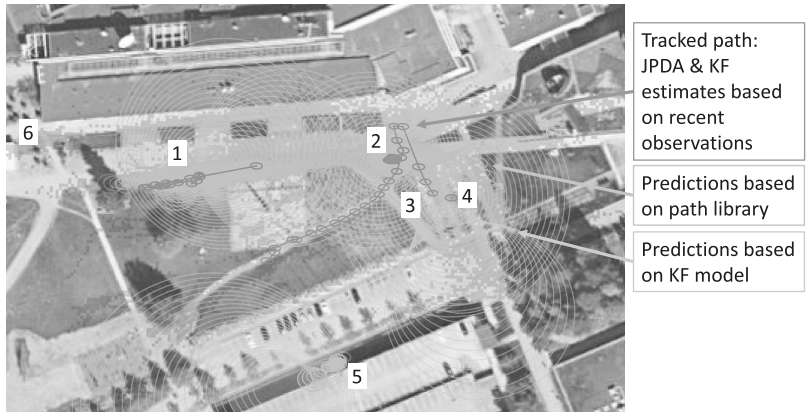


Figure 8: Real data example: path tracking and predictions with KF and path library.

not expand to as large area as the uncertainties of the KF predictions but sometimes divide into different branches.

To get more focused comparisons, we computed the short-, middle-, and long-term predictions as described

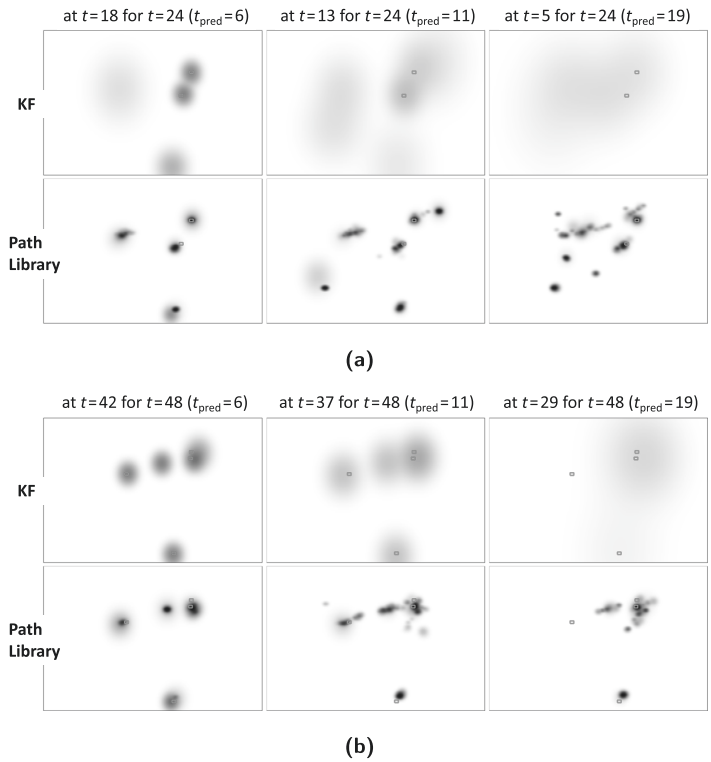


Figure 9: Real data examples: comparing area occupancy predictions by KF (top) and path library (bottom). The red markers give the locations of objects at the end of prediction: at $t = 24$ s in Example (a) and at $t = 48$ s in Example (b). The gray clouds indicate the predicted probability of the location being occupied by the agent. The predictions were made using observations obtained 6 s (left), 11 s (middle), and 19 s (right) earlier, respectively. (a) Predictions for $t = 24$ s. (b) Predictions for $t = 48$ s.

in Section 4.3. The occupancy predictions plotted on the maps together with the detections are shown in Figure 9, where predictions for two time instances are given as examples. In the occupancy map, the darker the color of a pixel, the more probable it is to detect an object in the pixel. It can be seen that with the both prediction methods, the accuracy of the prediction gets "diluted" as the prediction time increases. However, the dilution appears in different ways with the two methods. While the predicted occupancy areas of the KF get lighter and spread over large areas, the library-based predictions get an increasing number of smaller, more condensed occupancy patches. The examples of occupancy area predictions show that the path library gives much more accurate, but multimodal predictions. The differences become larger with increasing prediction times and they are clearly visible already with prediction length 11 s.

6 Discussion

The main contribution of this article consists of prediction of the future paths of mobile objects while preserving the privacy of the tracked objects. To improve the predictions, we proposed a method based on the library of paths tracked and recorded in the past. We demonstrated the operation of the library-based prediction using privacy preserving data obtained with an inexpensive computer vision system. With the same data, we compared the library-based predictions with KF-based predictions.

The requirement of privacy preservation in the data processing poses challenges to the path prediction by bringing on the need to solve the DA problem. This applies to both the initial state estimation and the collection of the path samples into the library. Despite the promising results of the library-based prediction, DA errors in the tracking are possible. They may happen when the paths of two (or more) objects coincide closely, which is possible, e.g., when the paths cross each other, or the paths coincide in a turn, or the paths evolve closely in the same direction with the same speed. In general, we do not consider the DA errors as serious flaws for the proposed method as the method aims to answer the question "will there be anybody in certain location" rather than the question about who will be there. However, DA errors could produce inaccurate initial state estimation or cause some wrong transitions between path segments to be learned to the library. We assume the statistical weighting in the usage of the library will mitigate the effect of these errors.

Although the model parameters σ_w^2 and R (defined in Section 3.2.1) were chosen using a general knowledge about what could be possible for a pedestrian rather than by optimization and fitting to the data, the library-based prediction performed surprisingly well in the demonstration with real data. This suggests the good robustness of the model.

Using low-cost, consumer grade equipment contributes to the inaccuracy and uncertainty of the initial state estimates for the predictions and the library paths and the need for the extra complexity of the timing scheme presented in Figure 7 for the performance evaluation of the prediction method. Although the proposed prediction method does not require the use of inexpensive equipment, the good performance with such in the demonstration suggests the robustness of the method. However, upgrading the equipment to a more expensive, higher quality vision system would allow more accurate tracking and to some extent, decrease the probability of DA errors.

For the applicability of the library-based prediction in changing environments, such as construction sites, the path library may require a forgetting mechanism to give smaller weight in the prediction to older paths that do not have recent examples. To improve the scalability of the path library, i.e., to reduce its resource requirements regarding memory and search times as the number of paths in the library grow large, the instance-based structure of the library could be replaced with a model-based structure that improves the compression of the stored path information. For instance, path segments with similar speed profiles and located close together could be combined to one, and long paths could be split to shorter "path primitives," e.g., links that connect nodes where the paths typically branch off or join together.

7 Conclusions

In this article, we propose a path library-based method to predict future paths of mobile objects. The predictions are based on the library of the observed past paths in the area and a short initial path segment estimated from the coordinates of the most recent object detections from an inexpensive, privacy-preserving vision system. We compared the library-based predictions to the predictions based on the KF combined with joint probabilistic DA. The performed tests show that the path library gives much more accurate but multimodal predictions. The difference increases with longer prediction lengths, and in the presented examples, it was significant already with prediction lengths of 11 s. However, despite its weaker prediction capability, the KF is needed in the library-based prediction as a fallback method in situations when

prediction is needed to areas that are not covered by examples in the path library and as a preprocessing stage to track the initial path needed for the computation of the library-based prediction. The directions of future development of the prediction method could include improving the positioning accuracy of the vision system by a wide-baseline stereo camera and depth estimation capability. An obvious target for further research is also the optimization of the model parameters.

Funding information: This work was partially funded by Business Finland project 408/31/2018 MIDAS.

Conflict of interest: Authors state no conflict of interest.

Data availability statement: The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

[1] Misra P, Enge P. Global positioning system: signals, measurements, and performance. 2nd edn. Lincoln, Mass: Ganga-Jamuna Press; 2006.
[2] Seppänen M, Ala-Luhtala J, Piché R, Martikainen S, Ali-Löytty S. Autonomous prediction of GPS and GLONASS satellite orbits. Navigation. 2012;59:119–34.
[3] Schöller C, Aravantinos V, Lay F, Knoll AC. What the constant velocity model can teach us about pedestrian motion prediction. IEEE Robotic Autom Lett. 2020;5:1696–703.
[4] Russell SJ, Norvig P. Artificial intelligence: a modern approach. 2nd edn. Upper Saddle River, New Jersey: Prentice Hall; 2003.
[5] Yrjanainen J, Ni X, Adhikari B, Huttunen H. Privacy aware edge computing system for people tracking. In: 2020 IEEE International Conference on Image Processing (ICIP); 2020. p. 2096–100.
[6] Redmon J, Farhadi A. YOLOv3: an incremental improvement. arXiv, 2018.
[7] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, et al. SSD: single shot multibox detector. In: European Conference on Computer Vision; 2016. p. 21–37.
[8] Lin T, Goyal P, Girshick RB, He K, Dollár P. Focalloss for dense object detection. arXiv preprint arXiv:1708.02002; 2017.
[9] Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. In: International Conference on Neural Information Processing Systems (NIPS'15) – Volume 1. Montreal, Canada: MIT Press; 2015. p. 91–9.
[10] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2014. p. 580–7.
[11] He K, Gkioxari G, Dollár P, Girshick RB. Mask RCNN. arXiv preprint arXiv:1703.06870. 2017.
[12] Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft coco: common objects in context. In: European Conference on Computer Vision; 2014. p. 740–55.
[13] Brown R, Hwang P. Introduction to random signals and applied Kalman filtering. 3rd edn. New York: John Willey & Sons Inc.; 1997.
[14] Bar-Shalom Y, Li X-R. Estimation & tracking: principles, techniques and software. Storrs, CT: YBS Publishing; 1998.
[15] Syrjärinne J. Studies of modern techniques for personal positioning. Ph.D. thesis, Finland: Tampere University of Technology; 2001.
[16] Bar-Shalom Y, Daum F, Huang J. The probabilistic data association filter. IEEE Control Syst Magazine. 2009;29:82–100.
[17] Luetteke F, Zhang X, Franke J, Implementation of the Hungarian method for object tracking on a camera monitored transportation system. In: ROBOTIK 2012; 7th German Conference on Robotics; 2012. p. 1–6.
[18] Sahbani B, Adiprawita W. Kalman filter and iterative-Hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system. In: 2016 6th International Conference on System Engineering and Technology (ICSET); 2016. p. 109–15.
[19] Adhikari B, Peltomaki J, Puura J, Huttunen H. Faster bounding box annotation for object detection in indoor scenes. In: 7th European Workshop on Visual Information Processing (EUVIP); 2018. p. 1–6.