DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

UNIVERSIDAD DE GRANADA

PROGRAMA DE DOCTORADO EN TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN

METODOLOGÍAS DE DATOS DE CALIDAD (SMART DATA) PARA DEEP LEARNING: EL PROBLEMA DEL RUIDO DE CLASE Y APLICACIONES EN CORALES Y COVID-19

Memoria presentada por

ANABEL GÓMEZ RÍOS

DIRECTORES

FRANCISCO HERRERA TRIGUERO

JULIÁN LUENGO MARTÍN

Editor: Universidad de Granada. Tesis Doctorales
Autor: Anabel Gómez Ríos
ISBN: 978-84-1117-464-0
URI: http://hdl.handle.net/10481/76794

Esta tesis doctoral ha sido desarrollada con la financiación de una beca de Formación de Profesorado Universitario concedida en 2016 a Anabel Gómez Ríos por el Ministerio de Educación, Cultura y Deporte con código FPU16/04765, y por los proyectos nacionales TIN2017-89517-P, concedido por el Ministerio de Economía y Competitividad, y PID2020-119478GB-I00, concedido por el Ministerio de Ciencia.

El conocimiento científico pertenece a la humanidad.

ALEXANDRA ASÁNOVNA ELBAKIÁN

AGRADECIMIENTOS

El desarrollo de esta tesis doctoral y su posterior culminación en esta memoria no hubiese sido posible sin el apoyo incondicional de mis directores de tesis, familiares y amigos.

Me gustaría empezar agradeciendo a mis directores, Julián y Paco, por su guía en este camino y por el tiempo, paciencia y dedicación que me han brindado durante mi formación como investigadora. Sin su trabajo y ayuda, esta tesis no sería lo que es hoy.

Quiero agradecer en especial a mi familia, sin cuyo apoyo y amor yo no habría podido llegar hasta aquí. A mis padres, Juan y María, por darme la fuerza para seguir, por alegrarse conmigo, incluso más que yo, con mis logros y buenos momentos y por acompañarme, siempre, durante los malos momentos. También por enseñarme, desde pequeña, la importancia de mis estudios, mi trabajo y mi tiempo. A mis hermanos, Juan Pedro y Elena, por aguantarme, por alegrarme y animarme cuando me hacía falta y, en definitiva, por hacerme saber que siempre puedo contar con vosotros. A mi pareja, Jacinto, por ser un apoyo indispensable durante estos años. Eres la persona que más me ha aguantado y ayudado a seguir. Estoy segura de que esto no habría salido adelante sin todos vosotros, por lo que esta memoria es vuestra también. Gracias por todo.

También me gustaría agradecer a todos los amigos y compañeros que me han acompañado durante estos años. A mis amigos de la infancia y adolescencia, por haber estado siempre ahí, aunque nos hayamos visto menos de lo que nos hubiera gustado. También a los que hice en la universidad, por todas las risas, por todo lo que hemos superado juntos y por aguantarme también en el proceso. Soy muy afortunada por teneros conmigo.

Por último, quiero agradecer a Ana, sin cuya ayuda y guía esta memoria no habría sido posible.

ABSTRACT

Currently, all the processes that are being executed in governments, companies and research centres are generating data that will be processed to extract valuable information. The process of extracting relevant information in data is known as Knowledge Discovery in Databases. This process contains two important steps, which are data cleaning and preprocessing, and data mining. The first one cleans the data in terms of inconsistencies, possible missing values, noise (errors in the data), etc. The second one uses the clean or smart data generated in the first step and applies Machine Learning algorithms to extract patterns and information from the data.

Deep Learning, a branch of Machine Learning, is now being widely used due to its good performance, especially when the data is composed of images, even outperforming other Machine Learning algorithms. However, Deep Learning is known to need great quantities of data to perform well, which is a drawback for the application of Deep Learning algorithms in scenarios that lack a big volume of data.

In this thesis, we propose the use of different preprocessing and optimization techniques to be able to use Deep Learning, and in particular, Convolutional Neural Networks, when the image data sets that we have available are small (below 1500 images), because it is costly or hard to obtain more data. That way, we transform the small data sets into smart data that can be used to train Convolutional Neural Networks.

We focus on three main applications: the classification of coral species using underwater images, the diagnosis of COVID-19 using X-ray images and the accurate classification of small data sets using convolutional networks in the presence of label noise, which occurs when a subset of the images are misclassified in the available dataset.

In particular, we propose the following:

- For the classification of coral species based on underwater images, we encountered two main problems: the small size of the available data sets due to the need for expert biologists to label them, and that they contain very close-up patches of the corals that do not show the structure of the corals, which we called textures. In this line, we first study the application of transfer learning and data augmentation techniques to be able to train powerful Convolutional

Neural Network architectures with these data sets. Transfer learning allows us to start the training from a pre-trained state of the network because it has been pre-trained using other, much bigger in size, data set. As a result of this, we do not have to train the whole network and it is possible to classify the small data sets by only training two layers at the end of the networks. Data augmentation consists of applying transformations to the original images in order to obtain new, slightly different, images. These transformations can be rotations, zooms, changes in brightness or contrast, vertical or horizontal flips, which help the network to be invariant to that transformations. These two techniques have allowed us to obtain better results than previous proposals in this task. Additionally, we have created a new coral data set, that we have made public, containing images of the entire structures of the corals. With this new data set, we have tackled the classification of coral species by using either texture or structure images thanks to a new two-level classifier that we propose.

- For the diagnosis of COVID-19 based on chest X-rays we encountered one main problem, that was that the available data sets that were being used for this diagnosis were not suitable for this task for several reasons. The reasons include that the protocol to label an image as positive for COVID-19 is not made clear, that they contain mainly cases of severe COVID-19, which is easier to detect, and that they contain images from different sources, making the networks decide the diagnosis based on information that was outside of the lungs. As a solution, in conjunction with a team of expert radiologists, we create and made public a new data set without these problems that can be safely used for the diagnosis of COVID-19. Then, using this data set, we propose a methodology specially dedicated to the preprocessing and classification of this type of image, which includes three steps. In the first step, we remove the information outside of the lungs, forcing the networks to use the lungs to diagnose the disease. Then, we transform each image using two specific generators that boost the characteristics between having or not having the disease in the x-rays. Lastly, we used the transformed images to train a new convolutional network to decide if each original input image corresponds to a person that has the disease or not.

- For the accurate classification of small data sets using convolutional networks in presence of label noise, we develop a specific algorithm designed to help during the training process of the Convolutional Neural Network. Specifically, it uses the predictions of the network and the probabilities associated with

those predictions, during the training process, to remove and relabel the data. This process continuously cleans and polishes the data so that the network can improve its training and therefore obtain better results than the ones obtained without this algorithm. We test our approach using big data sets like CIFAR10 and CIFAR100 and compare it with other state-of-the-art proposals that only use this kind of data sets, but we also prove the usefulness of our approach for the previous coral data sets and COVID-19 data set, as label noise can be even more harmful to smaller data sets.

The favourable results we have obtained in these three applications or tasks validate the use of Convolutional Neural Networks when the problems we want to resolve do not allow us to obtain bigger data sets. We have proven that even in these cases, when using appropriate preprocessing techniques, Deep Learning outperforms the results of other Machine Learning algorithms.

RESUMEN

Actualmente, todos los procesos que son ejecutados en gobiernos, empresas y centros de investigacións están generando datos que serán procesados con el objetivo de obtener información de valor. El proceso de extraer esta información relevante en los datos es conocido como *Knowledge Discovery in Databases*. Este proceso contiene dos pasos importantes, conocidos como limpieza y preprocesado de datos, y *data mining*. El primero limpia los datos originales en términos de inconsistencias, posibles valores perdidos, ruido (que son pequeños errores en los datos), etc. El segundo usa este conjunto ya limpio generado en el primer paso y usa algoritmos de aprendizaje automático para extraer patrones e información de estos datos.

El *Deep Learning*, una rama del aprendizaje automático, está siendo ampliamente usado ahora debido al buen rendimiento que ha mostrado, especialmente cuando los datos de entrada están compuestos por imágenes, superando los resultados obtenidos por otros algoritmos de aprendizaje automático. Sin embargo, los algoritmos de *Deep Learning* son conocidos por necesitar grandes cantidades de datos para obtener buenos resultados, lo que supone un inconveniente para su aplicación en escenarios que carecen de un gran volumen de datos.

En esta tesis, proponemos el uso de distintas técnicas de preprocesamiento y optimización que nos permitan el uso de algoritmos de *Deep Learning* y, en particular, redes neuronales convolucionales, cuando los conjuntos de datos de los que disponemos son pequeños (con un tamaño por debajo de las 1500 imágenes) debido a que es costoso y difícil obtener más datos. De esta forma, transformamos estos conjuntos pequeños en lo que se conoce como *smart data*, para que puedan ser usados para entrenar redes neuronales convolucionales.

Nos centramos en tres aplicaciones principales: la clasificación de especies de coral usando imágenes tomadas bajo agua, el diagnóstico de COVID-19 usando radiografías y la clasificación precisa de pequeños conjuntos de datos cuando contienen ruido de clase, lo que ocurre cuando un subconjunto de los mismos está mal clasificado como otra clase.

En concreto, nuestras propuestas son:

- Para la clasificación de especies de coral basada en imágenes tomadas bajo agua, nos encontramos con dos problemas principales: que los conjuntos de datos

disponibles eran pequeños debido a la necesidad de que expertos biólogos etiqueten las nuevas imágenes, y que éstos contienen parches de los corales que están tomados muy de cerca, de forma que las imágenes no muestran la estructura completa de los corales, a las que llamamos texturas. En esta línea, primero estudiamos la aplicación de las técnicas *transfer learning* y *data augmentation* para ser capaces de entrenar redes neuronales convolucionales con estos conjuntos de datos. El *transfer learning* nos permite empezar el entrenamiento de la red desde un estado de preentreamiento previo, debido a que la red ha sido preentrenada en otro conjunto de datos mucho mayor. Gracias a esto, no es necesario que entrenemos la red completa y es posible clasificar los conjuntos de datos pequeños entrenando sólo las dos últimas capas de la red. El *data augmentation* consiste en usar transformaciones sobre las imágenes originales para obtener nuevas imágenes, ligeramente diferentes. Estas transformaciones pueden ser rotaciones, ampliaciones, cambios en la luminosidad o el contraste, volteos verticales u horizontales, que ayudan a la red a ser invariante frente a estas transformaciones. Estas dos técnicas nos han permitido obtener mejores resultados que otras propuestas en esta tarea. Adicionalmente, hemos creado un nuevo conjunto de datos, que hemos hecho público, que contiene imágenes de las estructuras completas de los corales. Con este nuevo conjunto, hemos llevado a cabo la clasificación de especies de coral usando tanto imágenes de textura como de estructura, gracias a un nuevo classificador de dos niveles que hemos diseñado.

- Para el diagnóstico de COVID-19 basado en radiografías pectorales nos encontramos con un problema principal, que fue que los conjuntos de datos que estaban disponibles y que estaban siendo usados para esta diagnosis no eran adecuados para llevar a cabo esta tarea por varias razones. Estas razones incluyen que no se conoce el protocolo para etiquetar una imagen como positiva por COVID-19, que estos conjuntos contienen en su mayoría casos de COVID-19 muy severo, que es más fácil de detectar, y que contienen radiografías de diferentes fuentes, haciendo que las redes decidan la diagnosis basándose en información de las radiografías que estaba fuera de los pulmones. Como solución, junto con un equipo de expertos radiólogos, creamos e hicimos pública una nueva base de datos que no tiene estos problemas y que puede ser usada para la diagnosis de COVID-19. Después, usando este conjunto de datos, proponemos una metodología especialmente diseñada para el preproce-

samiento y clasificación de este tipo de imágenes, que incluye tres pasos. En el primer paso, eliminamos la información de las rafiografías que está fuera de los pulmones, forzando a las redes a usar la zona de los pulmones para realizar el diagnóstico. Después, transformamos cada imagen usando dos generadores específicos diseñados para potenciar las características de las radiografías que hacen que se clasifiquen como positivas o negativas de COVID-19. Finalmente, usamos las imágenes transformadas para entrenar una nueva red convolucional que decide si cada imagen original de entrada corresponde a una persona que tiene la enfermedad o no.

- Para la clasificación precisa de pequeños conjuntos de datos usando redes convolucionales cuando éstos presentan ruido de clase, desarrollamos un algoritmo específico diseñado para ayudar durante el proceso de aprendizaje de la red convolucional. Específicamente, usa las predicciones de la red y las probabilidades asociadas con esas predicciones, durante el proceso de aprendizaje, para eliminar o reetiquetar las imágenes del conjunto de datos. Este proceso limpia y perfecciona el conjunto de datos de forma continua de forma que la red pueda mejorar su entrenamiento y por tanto obtener mejores resultados que aquellos obtenidos sin este algoritmo. Probamos que este enfoque es válido usando grandes conjuntos de datos como CIFAR10 y CIFAR100 y lo comparamos con otras propuestas del estado del arte que sólo usan este tipo de bases de datos, pero también probamos la utilidad de nuestra propuesta con los conjuntos de datos previos de especies de coral y COVID-19, dado que el ruido de clase puede ser aún más perjudicial en conjuntos de datos pequeños.

Los resultados favorables que hemos obtenido en estas tres aplicaciones avalan el uso de redes neuronales convolucionales cuando los problemas que queremos resolver no nos permiten obtener conjuntos de datos más grandes. Hemos probado que incluso en estos casos, cuando usamos técnicas de preprocesamiento adecuadas, el *Deep Learning* mejora los resultados obtenidos por otros algoritmos de aprendizaje automático.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

PHD DISSERTATION

1 INTRODUCTION

Nowadays, every little process carried out in every company, research centre or government is generating data that is being recollected with the intention to process it in order to extract valuable information. The information extracted from the data is very important from the point of view of the companies, research centres and governments. The amount of data generated and collected is too large to be analysed by hand, so it is analysed using computers via a process called *Knowledge Discovery in Databases* (KDD) [PF91; HPK11]

The KDD process is composed of different steps that allow the extraction of valuable information or patterns in the data. They include the following:

- Problem specification: it includes an in-depth specification of the problem and its objectives.
- Data selection and sampling: it is the process of extracting the relevant data to the problem from the database or databases.
- Data cleaning and preprocessing: it consists of removing inconsistencies from the data and then preprocessing it [GLH15] (which can include noise removal [FV13], imputing missing values [LGH12], removing redundancies, etc.). This step creates what is now known as *smart data* [GRL+16], and allows the data to be used in the following step.
- Data mining: it consists of the application of *Machine Learning* (ML) algorithms. ML is the family of algorithms that are responsible for extracting relevant patterns given the curated data.
- Result interpretation: it is the process of interpreting the patterns extracted in the previous step, usually using different visualizations.

Although the most recognisable step is data mining, so much so that it is even used as a synonym for the entire KDD process [HPK11], the cleaning and preprocessing step is essential and equally important. This step produces quality data (smart data),

which is necessary for the data mining step to produce quality results and relevant relations in the data. Without cleaning and preprocessing the raw data, we could obtain erroneous relationships and results.

Commonly, data mining algorithms are classified into three main categories, depending on the available information of the target variable that we want to predict. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be the curated input data, where we have n instances (also called examples), and $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ the target variable for each of these instances. Each one of the variables in the data set is also called a characteristic. Then, the three main categories of data mining algorithms are the following:

- Supervised learning [CCD08], where the target variable $\mathbf{y}$ is known. The algorithms in this type of learning infer relations and patterns between the known data $\mathbf{X}$ and the target variable $\mathbf{y}$, so they can be used to predict the target variable of new, unseen data. There is a more detailed classification, depending on the domain of the target variable:
  - Classification, where the domain of $\mathbf{y}$ is discrete and we know all the possible classes or categories for each instance in the data: $y_i \in \{1, 2, \ldots, K\}$, $\forall i = 1, 2, \ldots, n$ and K is the total number of classes in the problem. If $K = 2$, it is usually called binary classification, and if $K > 2$, multi-class classification. For example, if we want to predict if a patient has COVID-19 or not.
  - Regression, where the target is continuous $y_i \in \mathbb{R}, \forall i = 1, 2, \ldots, n$. An example of regression is predicting the temperature of an industrial oven.
- Unsupervised learning [HTF09], where the target variable $\mathbf{y}$ is not known. The algorithms in this type of learning find relations and patterns between instances of the known data $\mathbf{X}$. We can also differentiate two main categories inside unsupervised learning (though there are more, like anomaly detection):
  - Clustering, which identifies relations in the data based on a similarity metric and creates groups of instances in a way that the similarity between instances of the same group is higher than the similarity between instances of different groups.
  - Association rules, where the algorithms identify relations between the data variables or attributes, generating a list of rules that describe the data.

- Semi-supervised learning [ZG09], where the target variable $\mathbf{y}$ is known only in a subset of the data available during the training process, but not in the rest. For example, a lot of medical applications work with semi-supervised learning, because it is necessary to have experts to label the data set, which is hard and costly to do for the entire data set.

This thesis focuses on supervised classification, where we have a *classifier* that is trained using an ML algorithm. The original data set is divided into a *training set*, used to train the classifier (where the classifier identifies patterns in the training set and learns to classify them), and a *test set*, used to test the effectiveness and generalisation ability of the classifier. The test set is not used during the training process. To test this we used the *accuracy* metric, defined as the number of instances of the test set correctly classified by the classifier over the total number of instances of the test set. That way, the accuracy gives us the level of confidence of the classifier when classifying new, unseen samples.

In the context of supervised classification, we focus on improving the accuracy of different classifiers by combining the use of optimization techniques (such as *transfer learning*, which we will see later) with the improvement of the data itself, in order to obtain what is called smart data, by using different preprocessing techniques. In this line, this thesis has three defined branches: the accurate classification of coral species using coral underwater images, the accurate classification of the COVID-19 disease using X-ray images and the accurate classification of image data sets under *label noise*.

Label or class noise [GLH15; FV13] refers to a specific problem in the input data set where the labels of some of the instances are wrong, that is, the label y_i of some instances of the data set does not correspond to the original class of the instances. The percentage of wrong labelled instances is called the *noise rate*, though this rate is usually not known in real-world data sets. This noise causes the ML algorithms to learn non-real patterns, and as a consequence, the algorithms do not have good generalization ability and they perform worse on unseen data. Therefore, it is crucial to use algorithms that can deal with label noise, either by preprocessing the training data trying to remove or correct the noise or by creating robust algorithms that are not affected by the label noise. In this context, we want to make a robust algorithm that is able to deal with label noise when the input data are images.

The three branches in this thesis have in common that each instance or sample in the data sets that we work with is an image. As a consequence, and since *Deep Learning* (DL) has shown an increased performance when dealing with images, we do not use

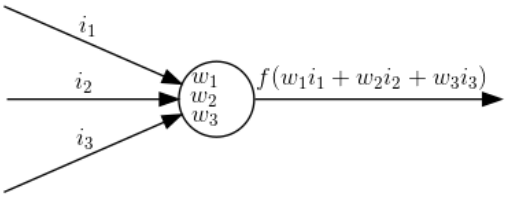


Figure 1.1: Example of a neuron in a deep neural network.

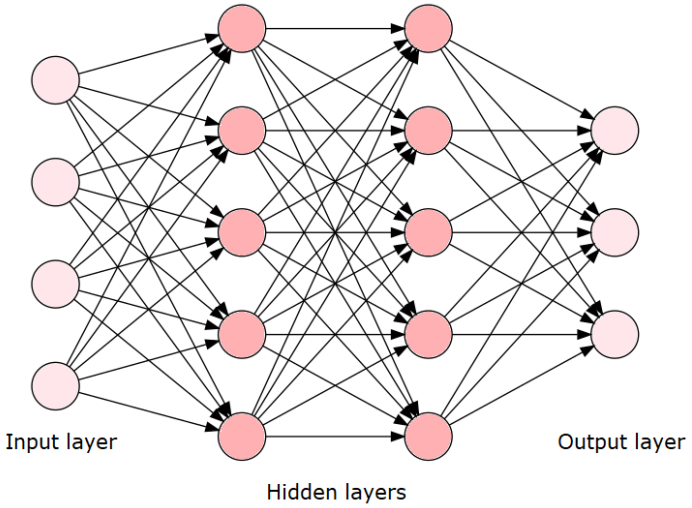


Figure 1.2: Example of a feedforward neural network.

classical ML algorithms like support vector machines or random forest. Instead, we choose to use DL, and, in particular, *Convolutional Neural Networks* (CNNs).

Deep Learning is a subclass of Machine Learning. The algorithms that belong to the DL family have in common that they are able to use complex representations by combining a lot of simpler representations and non-linear transformations [GBC16]. These algorithms have a specific structure called *network*, which is composed of *artificial neurons*, mimicking the structure of human neurons. The artificial neurons are nodes that are composed of three elements: 1) one or more inputs; 2) one output; 3) intrinsic weights. In Figure 1.1 we can see an example of a neuron inside a deep neural network. We can see that each neuron outputs a function over the inputs

and its weights $f(i_1w_1 + \dots i_nw_n)$, where $i_j, j = 1, \dots n$ are the inputs, n is the total number of inputs and w_j are the weights. These neurons are then combined forming layers that are stacked onto each other forming networks. Depending on the structure of the network and the functions performed in them, we can have feedforward neural networks, such as CNNs, or recurrent neural networks, such as the *Long Short-Term Memory* (LSTM) networks. In Figure 1.2 we can see an example of a feedforward network, where all the connections in a layer go to the following layers, not previous layers nor to the same layer. It is also a fully connected network because all the neurons in a layer are connected to all the neurons in the previous layer. In this thesis, we focus on CNNs, which owe their name to the main operation they perform: the convolution, which we will explain in the next chapter. These types of networks are perfect to use with images as data input because they can deal with matrices, and, in fact, the input of each layer of the network, not just the input, is multi-dimensional.

Traditionally, DL has been characterized as needing a big quantity of data to be able to perform well. While this is true, there are techniques, such as *transfer learning* and *data augmentation*, which we will see in-depth in the next chapter, that allow us to use DL algorithms when we are dealing with small data sets. In some problems and situations, it is fairly difficult to obtain a great number of images that are also of high quality. By quality in an image we mean two main things: 1) quality in the sense of how well the image looks: that it is not blurred, out of focus or too bright or dark to see the main object in the image; 2) quality in the sense of how well is the data set labelled, i.e. free of label noise.

When dealing with images as input data, the classical concept of machine learning classification can be further divided into other three concepts:

- Classification of the entire image, where we assign a label to each complete image. Usually, we want to classify the predominant object in the image.
- Detection, where we assign one or various layers to different parts of each image. Here, we want to detect the smallest rectangle that contains an object and classify the object in that region.
- Segmentation, where we want to divide the pixels of the image into regions that correspond with different objects detected in the image.

In this thesis, we focus on the classification of the entire image, assigning a label to each complete input image. In particular, we want to analyse how well DL performs and which techniques we can use if the available data sets are *small*. In this sense,

the three data sets we used to classify coral species were all below 1500 images, and the data set containing X-rays of COVID-19 patients and patients without this disease was less than 1000 images. In both cases, it is difficult to obtain more images, either because you need special equipment to take underwater images and the exact locations to take them, as with the corals, or because you need special equipment to take X-rays and a team of expert radiologists combined with RT-PCR tests to label the new images. In addition, the time may be crucial, as happened with the COVID-19 diagnosis, so we could not wait until more images are added to the data set. To put the size of these data sets into context, the ImageNet data set [DDS+09], which was the widely used data set to train new CNN architectures due to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [RDS+15] competition, had more than 1.2 million images only for the training set.

In that way, the three branches of this thesis are focused on how to combine these small data sets with preprocessing techniques and methodologies for smart data in order to obtain good accurate classifiers to classify coral species and to detect the COVID-19 disease, even in the presence of label noise, when we can use the algorithm we created to deal with this problem.

This thesis is structured in two parts: the PhD dissertation and the publications associated with it.

The rest of the Part I is structured as follows: Chapter 2 contains the background concepts needed for this thesis. In Chapter 3 we justify the relevance of this thesis and the associated publications. In Chapter 4 we give the objectives we wanted to complete with this thesis. Chapter 5 describes the methodology we followed during the development of this thesis. Then, in Chapter 6 and Chapter 7 we give a summary of the results included in the thesis, along with the publications where they are contained, and a discussion on that results, respectively. Finally, Chapter 8 presents the main conclusions of this thesis and outlines the future work.

Then, Part II includes the four publications we developed for this thesis, organized in three groups, reflecting the three branches of the thesis:

1. The accurate classification of coral species, containing the following publications:
    a) Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation, in Chapter 9.
    b) Coral species identification with texture or structure images using a two-level classifier based on Convolutional Neural Networks, in Chapter 10.

2. The accurate detection of the COVID-19 disease, containing the following publication: COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on chest X-ray images, in Chapter 11.

3. The accurate classification of data sets in the presence of label noise, containing the following publication: A robust approach for deep neural networks in presence of label noise: relabelling and filtering instances during training, in Chapter 12.

2

PRELIMINARIES

This chapter presents the background concepts used in this thesis: computer vision in Section 2.1, DL and convolutional neural networks specifically in Section 2.2 and the label noise problem in Section 2.3.

2.1 COMPUTER VISION

Computer or artificial vision is a field that includes a wide range of methods to process images so that a computer can extract information from them [Sze10; GCM21]. This process includes the acquisition of the images, their processing and posterior analysis and the applications using computers [GCM21]. This also includes the analysis of the cameras used to take the images and their calibration, analysis of shading, texture and colour in the images and the mathematical methods that are heavily used in all the processes [FP11].

When artificial vision started in the 1970s, the focus was to detect and extract edges from the images in order to make computers understand them. Since then, a lot of different techniques were developed to extract features from the images to be able to feed them to a computer, like skin colour detection, intensity and shade variation detection, or long edge detection, among others. Usually, these features were used to reconstruct 3D scenes or to perform some changes in the images, such as image blending or deblurring images [Sze10]. In the 2000s the tasks of image classification, object recognition in images and image segmentation began to increase their popularity. These tasks obtained their maximum popularity in the 2010s with the incursion of Deep Learning.

The applications of computer vision vary from robotics to control industrial processes to autonomous vehicles or computer-human interactions. Two important applications are in biology and medicine. In biology, computer vision has been used to determine the quality of wood [ARV+17], to detect fires on forests [ZLZ+18], to detect diseases in the plants [LTJ21] or to distinguish species, like flower species [NZ08] or coral species [BEK+12]. In medicine, computer vision has been used to help experts to detect different types of cancer, like breast cancer [GWL+18] or lung

cancer [ANB17], or to diagnose other diseases, like Alzheimer's [FAA+17]. In this thesis we focus on one application in biology: the classification of coral species, and one application in medicine: the diagnosis of the COVID-19 disease.

2.2 DEEP LEARNING

As we stated in the previous chapter, DL architectures are networks that are composed of neurons. Each layer in the network performs a specific operation and thus, the networks can be seen as the composition of the functions in each layer. The parameters in each function are called weights and they are learned during the training of the network. In each step or epoch of the training process, the weights are modified to minimize the error or cost in each epoch. This error is the value, given the actual weights of the network and the input with its true output value, of the *objective function* (also called cost function or loss function). For regression, it is usual to use the mean square error or the mean absolute error as objective functions. For classification, it is usual to use the cross-entropy function.

Although the first DL architectures were defined in the late 1950s with the implementation of the perceptron, it was not possible to train them due to the computation capability of the computers at that time. It was not until 1958, with the proposal of the *back-propagation* algorithm [RDG+95], that it was possible to train networks with one or two hidden layers. It then became widely used and it still is, today, the algorithm used to train the deep neural networks. Briefly, the back-propagation algorithm has two phases, a first forward phase and a second backwards phase. In the first epoch of the training, the weights are randomly set. Then, in the following epochs, the algorithm works as follows. In the forward phase, the output of the network is calculated using the current weights. Along with the output, it calculates the error or loss of that output evaluating the objective function. Then, in the backwards phase, the algorithm calculates the partial derivatives of the objective function with respect to the weights in each layer, from the end of the network to the start. These gradients are then used in a gradient-based optimization algorithm, such as the Stochastic Gradient Descent (SGD), so the error gets minimized and the weights are changed accordingly, ready to start the next epoch of the training process.

However, more deep networks were not possible to train until the late 2000s, when two main things happened: the computation capability of computers continued to

grow and was already able to train deeper networks, and the size of the available data sets grew.

There are several types of deep neural networks. Some of the most known and used are feedforward networks, such as the multilayer perceptron, where there are no connections between neurons in the same layer nor previous layers; recurrent neural networks and Long Short-Term Memory (LSTM) networks in particular [YSH+19; GBC16], which are used to process sequential data and where the neurons can be connected with other neurons in the same layer or previous layers; autoencoders citecharte2018practical, which are artificial neural networks with a symmetric encoder/decoder structure; and CNNs, which we are going to see more in-depth in the next subsection, as they are the type of networks we have used in this thesis.

2.2.1 *Convolutional Neural Networks*

CNNs are a type of deep neural network where at least one of the layers is a convolutional layer. A convolutional layer is a layer that performs the *convolution* operation in its discrete form, which is defined, for a two-dimensional input and kernel, as [GBC16]:

$$C(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \ , \tag{2.1}$$

where I is the input of the convolution, K is the *kernel* of the convolution (also called *filter*) and C is the output of the convolution. The kernel contains the weights of the convolution, which will be learned during the training process. The size of the kernel is usually 1×1, 3×3, 5×5, 7×7 or 11×11, so it is smaller than the input. The latest network releases that have been proposed tend to use smaller kernel sizes, since concatenating convolutional layers with smaller kernel sizes serves the same purpose as one convolutional layer with a bigger kernel size, but in less time and computational cost. Since the convolution is commutative and the values of the kernel are going to be learned during the training of the network, we can flip the kernel in both directions and we can write:

$$C(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \ . \tag{2.2}$$

Equation 2.2 is actually another operation called *cross-correlation*, but we have seen that in this context, where the kernel is learned, they are equivalent. Since the

cross-correlation operation is simpler to implement, it is the one usually implemented in deep learning libraries.

Let's see an example using Equation 2.2. Suppose we have the following input and kernel:

$$I = \begin{pmatrix} 5 & 9 & 1 & 3 & 0 \\ 6 & 8 & 2 & 4 & 1 \\ 1 & 7 & 3 & 5 & 5 \\ 4 & 6 & 6 & 9 & 2 \\ 0 & 3 & 1 & 0 & 1 \end{pmatrix}, \quad K = \begin{pmatrix} 2 & -1 & 3 \\ 1 & -2 & 0 \\ 4 & 1 & -1 \end{pmatrix}. \tag{2.3}$$

Then we simply have to move the kernel through the input iteratively, taking neighbourhoods from the input of size 3×3, which is the size of the kernel, from top to bottom and left to right, multiplying the elements in the same position and summing the results, so the output will be:

$$C = \begin{pmatrix} 2 & 56 & 5 \\ 13 & 48 & 27 \\ -2 & 33 & 7 \end{pmatrix}. \tag{2.4}$$

Note that the output is smaller than the output. As bigger the kernel size, the smaller the output. To prevent this from happening, the convolution has an associated concept called *padding*, though it is not obligatory to use it. The padding consists of momentarily adding columns and rows to the input matrix until the output has the same size as the input. These newly added rows and columns could have zeros (which is known as *zero padding*) or can mirror the original values of the original matrix, for example.

There is another concept associated with the convolution, which is the *stride*. The stride controls the overlapping between the neighbourhoods we take from the input matrix when calculating the convolution. In the previous example, we used a stride of 1 since we moved one row or column at a time to obtain the next neighbourhood. However, a bigger stride could be used, causing the output matrix to be smaller in size.

Now, the definition we gave in Equation 2.1 and Equation 2.2 uses two-dimensional inputs and kernels, because it is easier to understand and describe in the example.

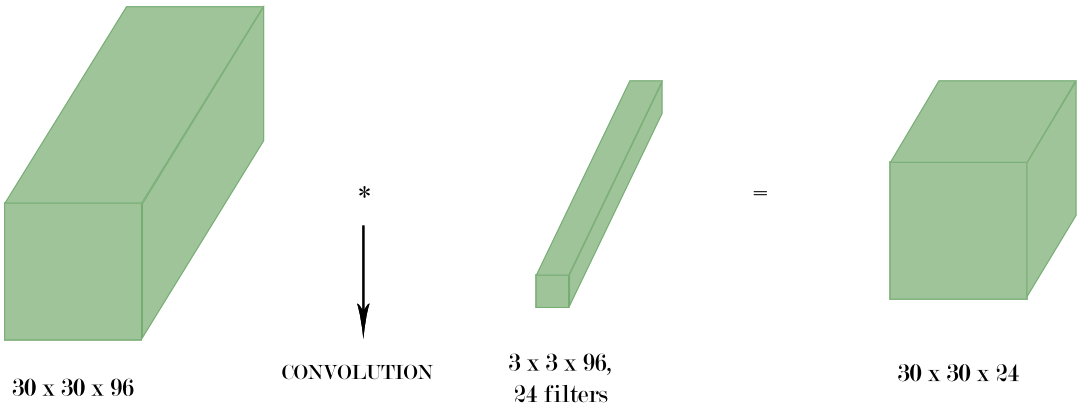


Figure 2.1: Example of a convolution performed on a convolutional layer, showing one of the 24 filters.

However, in an actual CNN, the convolutional layer performs a three-dimensional convolution. If, for example, the input of the CNN is a colour image we already have a three-dimensional input since we have the three channels R, G and B. But, moreover, each convolutional layer performs not one but several convolutions with the same input but different kernels, so the output of a convolutional layer (which will be at some point the input of another convolutional layer), has three dimensions, the third being the number of different kernels or filters we used in the previous convolutional layer. An example of an actual convolution in a convolutional layer can be seen in Figure 2.1.

Note that the third dimension of the kernels is not eligible and has to match the third dimension of the input. The first two dimensions of the kernels and the number of kernels in the convolutional layer are hyper-parameters of that layer and can be changed. Each one of the 24 matrices of size 30×30 of the output is called a *feature map*.

The non-linear function that characterize DL is usually introduced after each convolutional layer in CNNs, by a layer that performs that non-linear function element-wise. In CNNs the most used non-linear function is the *Rectified Linear Unit* (ReLU) function, defined element-wise as it follows:

$$\text{ReLU}(x) = \max(0, x) \ . \tag{2.5}$$

For example, if we take the output of the above example and perform the ReLU function, we obtain:

$$\begin{pmatrix} 2 & 56 & 5 \\ 13 & 48 & 27 \\ 0 & 33 & 7 \end{pmatrix} . \tag{2.6}$$

There are different non-linear functions, often variations of the ReLU function, like the Leaky ReLU, defined as x if $x > 0$ and $0.01x$ otherwise, or the Exponential Linear Unit (ELU), defined as x if $x > 0$ and $a(e^x - 1)$ otherwise, where a is a hyper-parameter that can be tuned. However, the most used non-linear function is the ReLU function.

Now, in a CNN we want to extract different characteristics through the depth of the network. To do this, the input size of the layers needs to be lowered, which can be done using a bigger stride in the convolutional layers, but it is more common to use another layer called *pooling layer*, whose aim is to downsample its input by using some mathematical operation that aggregates the inputs, like the average, the maximum, etc, and a stride greater than 1. In this case, we again take neighbourhoods of a predetermined size (which is a hyper-parameter of the layer) and perform the chosen operation. For example, if we perform a max-pooling with size 2×2 and stride 2 for the input I, we obtain the following output:

$$I = \begin{pmatrix} 2 & 56 & 5 & 0 \\ 13 & 48 & 27 & 7 \\ 0 & 33 & 7 & 23 \\ 14 & 19 & 5 & 2 \end{pmatrix} , \ \max \text{pooling}(I) = \begin{pmatrix} 56 & 27 \\ 33 & 23 \end{pmatrix} . \tag{2.7}$$

A CNN is then a concatenation of these layers. The way they are repeated and concatenated with each other determines the architecture of the network. In the first proposed architectures, such as LeNet-5 [LBD+89], the networks were a repetition of a convolutional layer, the non-linear activation and a pooling layer, though the networks only had a few layers. LeNet-5, for example, has five layers with weights. As the hardware improved and the computers were able to train deeper networks, more architectures were proposed, which started to stack convolutional layers on top of

each other, using fewer pooling layers, such as VGGnet [SZ14], which was proposed in 2014 with two variants: one with 16 layers with parameters and one with 19. Then the ILSVRC competition motivated a proliferation of new CNN architectures, which are deeper and more complex, like GoogLeNet [SLJ+15] (or its widely used newer version Inception v3 [SVI+16]), ResNet [HZR+16] or DenseNet [HLV+17]. These newer architectures are based on the repetition of a module (different in each case) that is repeated through the network and depending on the number of repetitions a different architecture is obtained. ResNet, one of the most used CNNs, has variants with 18 layers, 50 layers and 152 layers, among others.

As the depth of the networks began to increase, it became necessary to use other layers in order to be able to train them, such as normalization layers like *batch normalization*, or to alleviate overfitting, like *dropout*. Batch normalization [IS15] is a method of reparametrization that adaptively normalizes each feature map in its input to have zero mean and a standard deviation of 1, and then scales and shifts the result using two new parameters per feature map, allowing the network to learn the best mean and standard deviation for each input. Dropout [SHK+14] is a technique used in the training process that consists of cancelling out a percentage of the connections between neurons so that not all of the weights are learned in all of the training epochs. The connections that are cancelled in each epoch change and are selected randomly. Then, during the test process, all connections are used.

Finally, in the context of classification, we need the output of the networks to return a probability of each input image belonging to each class. To do this, the last layer of the network needs to be a *Fully Connected* (FC) layer, which is a one-dimensional layer where all its neurons are connected with all the neurons in the previous layer. The FC layer needs to have as many neurons as classes are in the problem we want to classify, and after that, it performs the *softmax* function, which is defined, from $\mathbb{R}^K$ to $[0,1]^K$, where K is the total number of classes, as:

$$\text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)} \ . \tag{2.8}$$

Then, the class of the input image is the one with the highest probability.

2.2.2 *Transfer learning*

Transfer learning is an optimization technique that allows starting the training process of a network from a pre-trained state, instead of starting with random weights on

all the layers of the network. This is critical when the data set we want to classify is small. We need two data sets, one data set A used to pre-train the network, and another data set B, which is the data set we want to classify. These two data sets are usually related in some way and the data set A is commonly much bigger in size. That way, we are transferring the knowledge obtained with data set A into data set B.

This technique is widely used because it speeds up the training process of data set B, but it is even more important in case data set B is small, because it would not have been enough to train the entire network from scratch.

Since the most used CNNs emerged thanks to the ILSVRC competition, they were already trained in the ImageNet data set [DDS+09]. The ImageNet data set has two main advantages: it is very big (1.2 million images) and contains 1000 common classes, such as different species of dogs, types of cars, buildings, etc. Another advantage is that the main deep learning libraries, like TensorFlow [MAP+15] and PyTorch [PGM+19], provide the most used CNN architectures with and without the pre-trained weights in ImageNet. That way, this step, which is long due to the size of ImageNet, is already done and we can directly retrain the networks on our data set.

The pre-trained networks on ImageNet will have one last FC layer with 1000 neurons (the number of classes in ImageNet), which we will need to replace with, at least, another FC layer but with as many neurons as classes are in the data set that we want to classify.

Once the networks are pre-trained and we have changed their last layer, we can choose to retrain the weights in all the layers of the network, some of them, or just the newly added layers at the end. This practice is called *fine-tuning*. Figure 2.2 shows an example of transfer learning where we are choosing to retrain some of the layers of the original network.

2.2.3 *Data augmentation*

Data augmentation is a technique used during the training process of a neural network, where the size of the training set is increased by performing several transformations to the original instances.

In our context, where the instances of the training set are images, these transformations are the following:

- Rotations of the images.
- Changes in the brightness of the images.

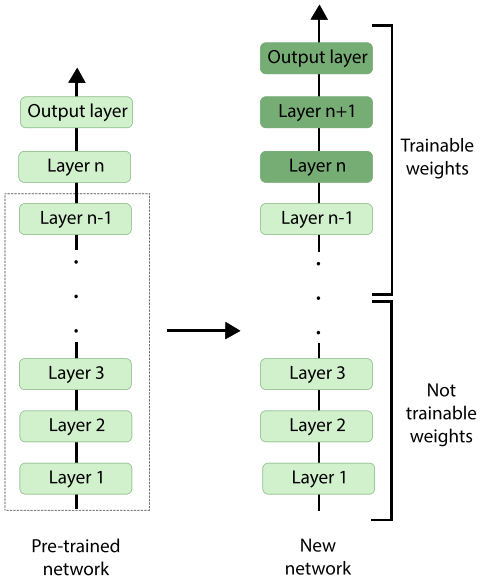


Figure 2.2: Example of transfer learning.

- Crops of the images.
- Flips, vertical and/or horizontal, of the images.
- Zooms of the images.
- Changes in the contrast of the images.
- Translations of the images.

These transformations are performed during the training process. TensorFlow, which is the framework we used throughout this thesis, chooses to apply them or not randomly for each image in every epoch of the training: for example, flipping or not an image vertically. Additionally, if the transformation implies some hyper-parameter, for example rotating an image, which implies a rotation degree, we can decide a maximum degree d to rotate them and TensorFlow chooses randomly a degree in the range $[-d, d]$.

The use of these transformations has the advantage that they made the CNN invariant to the transformation used, which is especially relevant if all the images in

the training set were taken under the same conditions because new unseen images are probably not going to be taken under the same conditions. But is also especially relevant if the original training set was small because it increases its size. However, we need to be careful with the transformations that we use in each case, because they can be harmful to the training, or make the images lose their meaning. For example, if we are trying to classify hand-written digits, as with the MNIST data set [Den12], we cannot flip the images, either vertically or horizontally, because they lose their meaning. We also need to be careful with the values we give to each transformation. In the above rotation example, if we use a too large maximum degree d, it can harm the training. As a result, these transformations need to be carefully checked for each problem, selecting which ones are suitable for the problem and then, which ones help the learning of the network.

2.3 LABEL NOISE PROBLEM

The problem of label or class noise occurs when there are instances in the available training set that are misclassified. There is a very common problem, which can have several causes: there were not enough experts to label the data set or maybe the data set was labelled automatically, which is a practice in deep learning, in order to obtain bigger data sets [XXY+15]. Sometimes, even if the data set is carefully labelled, we can still have some misclassified images. Therefore, it is extremely important to deal with these instances, because they can harm the training of the classifiers. In the specific case of DL, it has been studied that class noise damages the training of the networks [ZBH+21].

Several ways have been studied to deal with noise [GLH15]:

- Robust classifiers, which rely on the model to be less affected by the noise in the data set. In DL, this is usually done by modifying the loss function so that it can deal with the noisy samples [AWA+19; GKS17; ZS18], or by correcting its values [PRK+17; MWH+18].
- Data polishing methods, which clean the data before feeding it to the classifier by correcting the noisy samples [SKL19; PRK+17].
- Noise filters, which clean the data before feeding it to the classifier by removing the noisy instances [SH19; DWF+18].

The creation of good robust classifiers, data polishing methods or noise filters is critical for obtaining good results when classifying data sets that contain label noise. Robust classifiers are the most difficult to create, and while they can perform quite well if the noise rate is low, they could still be affected by higher noise rates [GLH15]. Data polishing methods can relabel instances, but they must detect very well the noisy instances, because they can change the label of an instance that was clean, increasing the noise in the dataset. In addition, it has to relabel the instances to their original class, and not another one, leaving the noise rate intact. Finally, noise filters cannot increase the noise in the data set, but they can reduce significantly the size of the training set if the noise rate is high or if it does not detect the noisy instances with precision, which aggravates the lack of data when dealing with small data sets.

There are various types of label noise that present different levels of difficulty when dealing with them [FV13]:

- Noisy Completely At Random (NCAR): the occurrence of a noise sample does not depend on the true class nor the sample itself. When the same percentage of instances are mislabelled in all classes, which is a common assumption, it is called symmetric noise.

- Noisy At Random (NAR): the occurrence of a noise sample depends on the true class of the instance but not on the instance itself. For example, if we suppose we have a problem with four classes: plane, bird, dog and house, we can assume that it is more probable that the images with planes and birds are confused with each other, than with the other classes. This type of noise is also called asymmetric label noise. It is more realistic than NCAR, so it is usually more complicated to deal with than NCAR.

- Noisy Not At Random (NNAR): the occurrence of a noise sample depends on the true class of the instance and on the instance itself. For example, for the COVID-19 problem, where we have a class P with the X-rays of patients who had the disease, and N with the X-rays of patients who did not have the disease, we can assume that is more probable that the patients in P who had a mild illness are more probable to be misclassified as N than the ones who were severely ill. This is the most realistic type of noise since it allows the instances more similar to the instances of another class to be the noisy ones. As a result, this type of noise is the most complex to deal with.

2.3.1 *The problem of label noise in the case of CNNs*

In the specific case of CNNs, most works focus on NCAR noise, usually called uniform or symmetric in this context [PRK+17; HMW+18; MWH+18]. Some of them also analyse the effect of NAR noise, called asymmetric noise in this context [JNC16; WMC+19; NMN+19]. However, to the best of our knowledge, the NNAR noise has not been tackled in this context.

There are several ways to tackle the noise problem when using CNNs. In general, we can distinguish the following:

- Methods that change the loss function, either to make it robust to the noise [WKH+19; PRK+17; HMW+18; MWH+18], or as a result of other changes in the network [SBP+14; SC18].
- Methods that model the noise distribution, either by previously estimating the noise matrix [PRK+17], by supposing it is known [SKL19] or by modifying the network to add a layer that models it [SBP+14; JNC16].
- Methods that correct the noise [SKL19; AOA+19].
- Methods that filter the noise [SKL19; NNL+19].

Note that this categorization is not exclusive, as some works use more than one approach.

We focused on creating an algorithm that filters and correct the noise during the training process and we tested it using the three types of noise (NCAR, NAR and NNAR) to see how well it will behave under different types of noise, especially the ones that are more realistic.

3

JUSTIFICATION

This chapter presents the open problems that justify this memory thesis.

As we stated in previous chapters, this thesis has three branches. We have approached all of them from the point of view of data quality or smart data and the need to use preprocessing techniques to be able to work with the data and obtain valuable information from it.

From this perspective, we show in this chapter the justification associated with the three lines of this thesis:

- The classification of coral species using CNNs. Most of the available coral data sets are small and they have the characteristic that the images are patches, that is, they are very close-up images that do not show the entire structure of the corals, just texture details. The state-of-the-art model used to classify coral species uses classic ML algorithms and these data sets. It is a very complex framework composed of nine steps and where each step is composed of several ML algorithms. The entire framework needs to be tuned to choose the best algorithm in each step for each data set. We proposed the use of CNNs in combination with transfer learning and data augmentation to simplify this process in a way that can be extrapolated to other problems with small data sets. In addition, we created a new data set that contains images of the entire structure of the corals and proposed a two-step classifier to accurate classify coral species based on texture or structure images. We made the new data set public.

- The diagnosis of COVID-19 based on X-ray images. In 2020, the COVID-19 disease was declared a pandemic and the diagnosis of patients with the disease was a number one priority. At that time, there were no rapid tests, and it was important to develop triage systems that were able to detect COVID-19 cases faster than a Reverse Transcription Polymerase Chain Reaction (RT-PCR) test. There was a data set of X-ray images that became very popular, but that presented several problems. In conjunction with a team of expert radiologists, we focused our work on three main lines. First, we analysed the available data

set and concluded that it was not suitable for the correct detection of the disease. Second, we developed and made public a curated data set of X-ray images that were suitable to detect the disease. And third, we developed a methodology which included several preprocessing steps in order to accurately diagnose if a patient had COVID-19 or not based on a chest X-ray.

- The accurate classification of data sets, in particular small data sets, in the presence of label noise when using CNNs. As we have seen in the previous chapter, the presence of label noise is detrimental to the training of CNNs. The vast majority of the available proposals in this line use big well-known data sets such as MNIST or CIFAR10/100, injecting symmetric and sometimes asymmetric noise. However, most times the proposals are specifically designed for these benchmarks and they are not tested in more realistic data sets, let alone small data sets. We proposed a robust approach for training CNNs that can be used with any CNN and we tested it in small data sets (in particular for the coral data sets and the COVID-19 data set) and in big data sets like CIFAR10 and CIFAR100. We injected the three types of noise (NCAR, NAR and NNAR) to test our approach under different scenarios.

4

OBJECTIVES

After addressing the open problems we wanted to tackle, in this chapter, we present the main objectives of this thesis. As we did in the previous chapter, we divide the objectives into the three branches of the thesis, which revolve around using different preprocessing and optimization techniques for obtaining smart data to obtain accurate classifiers. The realisation of this task involves an initial study of the problem, the design and implementation of the novel solution to the problem and the posterior evaluation of the proposed approach in conjunction with the comparison with the current state-of-the-art. The specific objectives that we propose in this thesis are the following:

1. *To propose an accurate classifier for coral species based on CNNs using underwater images, both texture images (close-up patches of the corals) and structure images (images containing the entire corals).* This includes the following objectives:
   1.1. A study of the current state-of-the-art in coral classification based on underwater images and the available data sets.
   1.2. The creation of a new data set containing structure images, which was not available.
   1.3. The study of different CNN architectures in order to obtain the best classifier for the texture problem and the structure problem.
   1.4. The study of transfer learning and data augmentation in these small data sets to improve the accuracy of the CNN classifiers.
   1.5. The proposal of a two-level classifier, using CNNs, which allows the classification of any coral image, either texture or structure.
2. *To propose an accurate classifier to diagnose cases of the COVID-19 disease using chest x-ray images of patients*, which includes the following objectives:
   2.1. A study of the available data sets and their suitability to solve this problem, along with a study of the already published proposals for this task.

2.2. The creation of a new quality data set, working together with a team of expert radiologists, which can be used to classify X-rays as positive COVID-19 or negative COVID-19.

2.3. The development of a new methodology to preprocess and classify X-ray images, which is composed of three steps. First, segmentation-based cropping is carried out to obtain the relevant part of the X-rays, since they contain information outside of the lungs. Then, we used a class-inherent transformation network based on generative adversarial networks, called FuCiTNet [RGT+20], to obtain two transformed images from each original image, one using a generator trained over the positive images and the other one using a generator trained over the negative images. Lastly, we used the ResNet50 architecture to classify each original image using its two transformations, also using transfer learning and data augmentation.

2.4 The comparison of our methodology with other proposals to resolve this task.

3. *To propose a robust approach for CNNs to train under label noise, especially when classifying small data sets*, which includes the following objectives:

3.1. The study of the proposals to help the training of CNNs under label noise that are already available.

3.2. The creation of a novel approach to deal with the three types of label noise: NCAR, NAR and NNAR.

3.3. To test our algorithm using small data sets, in particular, the coral data sets and the COVID-19 data set, and big data sets like CIFAR10 and CIFAR100. In addition, tos compare it with other state-of-the-art approaches.

5 METHODOLOGY

The methodology that we followed during the realisation of this thesis was an adaptation of the scientific method applied to the objectives of the previous chapter. Specifically, the methodology was the following:

1. *Observation*: an in-depth study of DL, and CNNs in particular, along with the label noise problem and the existent approaches for training CNNs in presence of label noise. In addition, studies of the state-of-the-art in the two applications in which we used CNNs: classification of corals and diagnosis of COVID-19.

2. *Hypothesis formulation*: the design of new methodologies to train CNNs with small data sets that contain label noise and methodologies to use CNNs in coral classification and diagnosis of COVID-19.

3. *Experimentation*: retrieving performance results of the three designed methodologies, measured in terms of the accuracy of the classifiers.

4. *Contrasting the hypothesis*: the comparison of the results we gathered from our classifiers with other state-of-the-art proposals, in each of the three branches of this thesis. These comparisons will serve to validate the effectiveness of our models.

5. *Hypothesis validation or refutation*: after the analysis of the results obtained, validation or refutation of the stated hypothesis. If rejected, the previous steps should be repeated, creating a new hypothesis to be proved.

6. *Scientific thesis*: Once the hypothesis is validated, the conclusions are made and the entire process is redacted into journal publications and this thesis.

6

SUMMARY

This chapter summarizes the proposed approaches in this thesis. We divide the chapter into three sections, each one containing one of the branches of the thesis. In each section, we present a brief description of the problem, the solution we gave to that problem and an outline of the experiments we carried out. Section 6.1 summarizes our approach for the classification of corals, Section 6.2 shows a brief description of our proposed methodology for the diagnosis of the COVID-19 disease and in Section 6.3 we summarize our proposal for the training of CNNs when the training set presents label noise. The discussion over the results we obtained will be presented in Chapter 7.

6.1 ACCURATE CLASSIFICATION OF CORAL SPECIES BASED ON UNDERWATER IMAGES

The automatic classification of coral species is an important task, as it can help experts to track endangered species in a faster way. Currently, there are Autonomous Underwater Vehicles (AUVs) taking photos of corals, but they are not being processed automatically because the classification of coral species entails several problems, such as the fact that the taxonomy is constantly changing as new species are discovered, that some coral species are very similar to each other, or the fact that some coral species usually appear together and it is difficult to take photos of them individually. In addition, underwater images present other kinds of problems, such as variations in lighting or blurring due to the water column between the camera and the coral.

There have been other works addressing the automatic classification of corals based on underwater images. Most of them use classical machine learning algorithms in combination with feature extraction methods [BEK+12; PRJ+08; SGG+13], and some of them use CNNs, but classical ones like VGGNet or LeNet [MBA+16a; MBA+16b; Ela15].

We proposed the use of more powerful CNNs, in particular Inception v3 [SVI+16], ResNet [HZR+16] and DenseNet [HLV+17] to classify two public small coral data sets: EILAT and RSMAS [Shi17], which contain coral patches or textures. EILAT contains

1123 coral patches distributed into eight classes and RSMAS contains 766 patches distributed into fourteen classes.

The state-of-the-art in these two data sets was a complex framework created by Shihavuddin et al [SGG+13], which uses classical machine learning models organized into nine steps. Inside each step, one or various machine learning algorithms can be used to first extract features from the images and then classify them. In addition to having to select the algorithms for each data set, each algorithm needs to be fine-tuned to each data set. Our proposal has fewer hyper-parameters to tune.

In addition, we created a new data set containing 409 structure images of various sizes that contain the whole structure of the corals and are distributed into the same fourteen classes as RSMAS. We named this data set StructureRSMAS.

This data set allowed us to classify coral species based either on their texture image or their structure image. In order to do that, we proposed a two-level classifier, composed of three classifiers. In the first level, we use a binary classifier that distinguishes whether an input image is a texture image or a structure image. Then in the second level, we have two classifiers, a texture classifier and a structure classifier. That way, if in the first level the binary classifier decides that the input image is a texture, the texture classifier is used in the second level. Equivalently, if the binary classifier decides that the input image is a structure in the first level, in the second level the structure classifier is used.

To compensate for the small sizes of the three coral data sets, we investigated the use of transfer learning from ImageNet and data augmentation in all cases. In particular, we only trained two newly added layers in each CNN, leaving the rest of the weights of the networks with their values pre-trained on ImageNet untouched. This speeded up the training process and allowed us to use such small data sets since we just have to train the weights of the last two layers. We used five-fold cross-validation and the accuracy metric to compare the results between the different models and types of data augmentation.

We also investigated the use of an intermediate transfer learning from another coral data set which contained more images, MLC-2008, fine-tuning all the weights on the CNNs as a previous step to train the last two layers with RSMAS and EILAT. Furthermore, as the data sets were not balanced, meaning that some classes had more images than others, we also analysed the use of a cost-sensitive loss function in order to improve the accuracy results.

In addition, we performed a comparison between our proposal to classify EILAT and RSMAS and the state-of-the-art, the framework from Shihavuddin et al [SGG+13],

and we improved their results. The two journal publications associated with this part are:

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A. S. M., Krawczyk, B., & Herrera, F. (2019). Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation. *Expert Systems with Applications*, 118, 315-328.

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A. S. M., & Herrera, F. (2019). Coral species identification with texture or structure images using a two-level classifier based on Convolutional Neural Networks. *Knowledge-Based Systems*, 184, 104891.

6.2 ACCURATE DIAGNOSIS OF COVID-19 BASED ON CHEST X-RAY IMAGES

In 2020, the coronavirus disease (COVID-19) was declared a pandemic by the World Health Organization (WHO). The diagnosis of the disease has been carried out using RT-PCR tests, Computed Tomography (CT) scans or chest X-rays. In 2020, the rapid tests were not available yet and the resources were limited, which made chest X-rays the fastest method to detect the presence of the disease. Moreover, the equipment used to obtain the X-rays is more lightweight than the one used for CT scans and could be transported to small hospitals where it was not possible to do CT scans and no RT-PCR tests were available.

Due to this, there were an increasing number of works using public chest X-ray images and DL to detect the presence of COVID-19 [WW20; AHN+20; OTY+20; KDR+20; NKP21]. However, most of these works used combinations of public chest X-ray data sets. The most popular one is COVIDx [WW20], which combined the COVID-19 image data collection [CMD20] with the RSNA pneumonia detection challenge data set from Kaggle [RSNA19] with another public collection of COVID-19 images [Chu20]. These public data sets have several problems. The first one is that the protocol followed for annotating an image as positive for COVID-19 is not made clear, and since most works combine several sources of images, they are probably merging different protocols. Furthermore, the images are very heterogeneous and they were taken using different types of equipment, introducing other sources of information in the X-rays besides the lungs. And more importantly, they use X-rays from children, but they only appear in the positive class for COVID-19. In this line, the work of

Maguolo and Nanni [MN20] shows that DL algorithms obtain the same accuracies when using the complete X-rays and when the lungs are removed from them. Lastly, and equally important, these data sets are biased towards severe COVID-19 cases, which are the easiest to detect, leaving mild and moderate cases aside.

As a solution, we proposed a new data set, named COVIDGR1.0, built with the collaboration of four expert radiologists from Hospital Universitario San Cecilio in Granada, using the same detailed protocol for all chest X-rays, that were taken in hospitals in Granada. Specifically, we assigned the label P (positive COVID-19) when the X-ray is taken within under 24 hours of a positive RT-PCR test, and the ground truth is the result of an RT-PCR test. In addition, all images were taken using the same equipment and all positive COVID-19 images were assigned a severity by the radiologists between Normal-PCR+ (asymptomatic, with a positive RT-PCR+ test), Mild, Moderate and Severe, so we could track that every severity is being tackled. We added the same number of negative (class N, no COVID-19) chest X-rays as we have of positives (class P), to keep the data set balanced. We made this data set available to the public.

Then, using our data set, we developed a methodology to classify it and diagnose the disease, named COVID-SDNet. This methodology is composed of three separate steps: 1) a segmentation-based cropping, 2) class-inherent transformations of the cropped images and 3) inference based on the transformations obtained in step 2). In the first step, we remove the parts of the X-ray that do not contain any part of the lungs using the U-Net segmentation model [Min20]. After using this model, we crop the X-rays to take the smallest rectangle that contains the segmentation, adding 2.5% of the pixels around it. In the next step, we use the FuCiTNet model [RGT+20]. This model is a Class-inherent Transformation Network based on Generative Adversarial Networks (GANs) that learns two generators, one per class, G_P and G_N, which learns the inherent-class transformations of the positive P class and the negative N class, respectively, that is, they learn the characteristics that bring each input image to its class. Then, we use these two generators two generate a P transformation and an N transformation of each input image. In the last step, we use the two transformations obtained for each input image to train a ResNet50 model and we design an inference step to combine the outputs of the two transformations to obtain the output class of the original X-ray.

Lastly, using our data set, we compared our methodology with two of the most popular approaches, COVIDNet [WW20] and COVID-CAPS [AHN+20], obtaining better results in terms of accuracy.

The journal publication associated with this part is:

Tabik, S., Gómez-Ríos, A., Martín-Rodríguez, J. L., Sevillano-García, I., Rey-Area, M., Charte, D., Guirado, E., Suárez, J. L., Luengo, J., Valero-González, M. A., García-Villanova, P., Olmedo-Sánchez, E. & Herrera, F (2020). COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on chest X-ray images. *IEEE journal of biomedical and health informatics*, 24(12), 3595-3605.

6.3 ROBUST APPROACH TO TRAIN CNNS IN PRESENCE OF LABEL NOISE

Label noise is a common problem when dealing with real-world data sets. DL and CNNs in particular are affected by this problem [ZBH+21], which is why is important to develop algorithms that help to train CNNs when the data set contains label noise.

The majority of proposals to help the training of CNNs under label noise are specifically designed to classify benchmarks like MNIST or CIFAR10/100 [PRK+17; AWA+19; GKS17; ZS18; MWH+18], and they are not tested for small real-world data sets or under all types of label noise. Moreover, some of them suppose some kind of information is known, such as the noise rate [SKL19].

We proposed an algorithm, called RAFNI (Relabelling And Filtering Noisy Instances), that does not suppose the noise rate is known and that we tested using real-world small data sets, such as the three coral data sets in Section 6.1 and the COVIDGR1.0 data set in Section 6.2.

The RAFNI algorithm can be used with any CNN as a backbone network, and it uses the predictions and their probabilities to filter or relabel the instances during the training process. RAFNI is based on the fact that during the first epochs of the training of a CNN, the clean instances are learned, while the noisy instances tend to have higher loss values. Then, the CNN starts to overfit the noisy instances. In fact, we can approximate the loss value of the instances in each epoch with a Gaussian Mixture Model (GMM).

RAFNI uses this knowledge and the GMM to implement two filtering mechanisms and one relabelling mechanism. The first filter mechanism uses a threshold over the loss values of the instances to remove from the training set the instances that have a loss value over that threshold. This threshold is dynamic and changes during the training process. The second filtering mechanism controls how many times an instance has been relabelled and removes from the training set the ones that have been removed too many times because they have a higher probability of being noisy.

Finally, the relabelling mechanism uses a threshold over the probabilities with which the backbone network predicts the labels of the instances, changing the label of an instance to what the backbone network predicts when the prediction probability exceeds this threshold. The threshold is also dynamic and changes during the training process.

We tested RAFNI with the coral data sets, the COVIDGR1.0 data set (where we removed the images with severity Normal-PCR+), CIFAR10 and CIFAR100. We used the CIFAR data sets to compare our algorithm with some of the state-of-the-art algorithms in this task.

The publication associated with this part is:

Gómez-Ríos, A., Luengo, J., & Herrera, F. (2022). A robust approach for deep neural networks in presence of label noise: relabelling and filtering instances during training. Submitted to IEEE Transactions on Neural Networks and Learning Systems.

7 DISCUSSION OF RESULTS

This chapter presents the discussions of the results we obtained in the publications associated with this thesis. We organize the discussions with the same sections as the previous chapter.

7.1 ACCURATE CLASSIFICATION OF CORAL SPECIES BASED ON UNDERWATER IMAGES

Regarding the classification of coral species, we performed a study using Inception v3, ResNet and DenseNet to search for the best classifier for the three data sets: EILAT, RSMAS and StructureRSMAS. We added two FC layers to the end of these networks, one with 512 neurons, followed by a ReLU function, and another one with as many neurons as classes in each data set, followed by the softmax function. As the data sets were small, we used transfer learning from ImageNet and trained only the two added layers, freezing the rest of the networks. For EILAT and RSMAS, we also tested an intermediate step, using another data set, bigger, called MLC-2008, to fine-tune all the layers in the CNNs before training the two added layers with EILAT and RSMAS. However, the transfer learning from ImageNet gave better results than the transfer learning from MLC-2008. We believe that this is due to the classes in MLC-2008, as four of its nine classes are not coral species.

We found that ResNet was the best CNN for the three data sets: ResNet50 for EILAT and StructureRMSAS and ResNet152 for RSMAS. Then, using the best model for each data set, we conducted a study on the use of different data augmentation techniques: translation or shift, zoom, rotation, flipping and combinations of these techniques. Though there was improvement using data augmentation, it was a slight improvement: around 1% of accuracy. We argue that this happens because of the nature of the images in these data sets: EILAT and RSMAS contain small images and very close-up, so the transformations used needed to be small, causing them to have little effect. In the case of StructureRSMAS, we argue that this can be explained by the excessive small size of the data set.

Then, we performed a study using a cost-sensitive loss function to tackle the imbalance ratios on EILAT and RSMAS. To do that, we multiplied the error of each instance by a factor that depends on the proportion of the images in the instance class with respect to the rest of the classes. That way, the loss function gives more importance to classifying images in smaller classes. However, the results obtained from using a cost-sensitive loss function were slightly worse than using the normal loss function. We argue that this is because the imbalance ratios on EILAT and RSMAS were relatively small: most of them are below four, and CNNs are tolerant to these levels of imbalance [BMM18].

When we compared our models with the state-of-the-art for EILAT and RSMAS, we found that we outperformed the other models, becoming state-of-the-art for these data sets. For StructureRSMAS, we found that the two-level classifier performed well: since the first level classifier obtained an accuracy of more than 99%, there was no loss in the accuracy in the second level of the model. In fact, there was a slight improvement when using the two-level classifier than when classifying RSMAS and StructureRSMAS separately.

7.2 ACCURATE DIAGNOSIS OF COVID-19 BASED ON CHEST X-RAY IMAGES

Regarding the diagnosis of COVID-19, we performed a study on the whole COVID-SDNet methodology, testing at each step if it improved the final classification or not. In this case, as we observed that we had variation in the results between two repetitions of the same experiment, we performed five different five-fold cross-validations and we compared the mean and standard deviation of all the results. We found out that each step of the COVID-SDNet methodology actually helped the final classification of the COVIDGR1.0 data set, obtaining the highest accuracy (and also the best balance between specificity and precision) when using the three steps of the methodology. In addition to being the best accuracy result, it was also the most stable result, as it had the smallest standard deviation.

When compared with the other two approaches, COVIDNet and COVID-CAPS, COVID-SDNet performed significantly better when using our curated data set. The main problem of COVIDNet was that its precision in the negative class was 3.36%, meaning that it classified almost everything as positive for COVID-19. COVID-CAPS did not have this problem, but the overall accuracy was worse than the one obtained by COVIDNet.

We also conducted a study of the accuracy of COVID-SDNet per severity level, and we obtained that the best accuracy was obtained by the most severe cases and it decreased as the severity of the disease decreased. This is the expected result, since the most severe cases showed more signs of the disease in the lungs, being easier to classify as positives.

Then, we studied the behaviour of the methodology when removing the positive images of Normal-PCR+ severity from the data set and the accuracy results improved since these images, despite being positive by RT-PCR, did not show signs of the disease in the lungs.

Finally, we performed an inspection of the model decisions using heatmaps and showed why our methodology classifies an image as positive or negative in the image itself, in order to understand the model. Our team of radiologists actually inspected these images and found that the model looked at the right regions of the X-rays to label an image as positive or negative, making our methodology a good triage system.

7.3 ROBUST APPROACH TO TRAIN CNNS IN PRESENCE OF LABEL NOISE

Regarding the training of CNNs in presence of label noise, we conducted several experiments. First, we tested our algorithm, RAFNI, with ResNet50 as the backbone network, against the backbone network alone. We used EILAT, RSMAS, StructureRSMAS, COVIDGR1.0-SN, CIFAR10 and CIFAR100. Between all data sets, we tested the three types of noise: NCAR, NAR and NNAR, and in each data set we tested several noise rates (between 4 and 8 noise rates, depending on the data set and the type of noise).

We found that RAFNI obtained better accuracies on all data sets at all noise rates, even at 0% noise in the case of StructureRSMAS, COVIDGR1.0-SN, CIFAR10 and CIFAR100. In general, the gain in accuracy that we obtained from using RAFNI increased as the level of noise increased, which is very good as it means that our algorithm performs well at low noise rates but also at higher noise rates.

Then, we used CIFAR10 and CIFAR100 to compare our algorithm with other state-of-the-art models. First, we compare RAFNI with algorithms that did not suppose the noise rate is known, using a Wilcoxon Rank-Sum test to compare them, and we found that our algorithm performs better with significant differences. Then, we studied the loss of our algorithm against a model that supposed the noise rate known, and we

obtained that, while the other algorithm was better, RAFNI performed quite well for CIFAR10 and asymmetric noise, even obtaining better accuracy when the noise rate is higher, which shows that RAFNI is a very good option when dealing with difficult types of noise.

8

CONCLUSIONS AND FUTURE WORK

8.1 CONCLUSIONS

In this thesis, we have tackled several problems and applications under a common objective: the study of the techniques and methodologies we can use over small data sets, to turn them into we what we have called smart data sets, to be able to successfully use CNN architectures to classify them.

In the first objective of this thesis, we focused on accurately classifying coral species using underwater images. We studied two primary techniques, transfer learning and data augmentation, which allowed us to train various CNN architectures with two small data sets. Then, we analysed the introduction of a different type of coral image, structure images, to further improve the classification of the coral species. The results of the two studies that we performed showed the importance of the transfer learning technique when dealing with such small data sets. The use of data augmentation, although it improves the results, it has less effect than initially expected, probably due to the characteristics of the images and the small sizes of the data sets.

For the second objective of this thesis, we proposed a methodology to pre-process and classify chest X-ray images with the objective of diagnosing the COVID-19 disease. We combined segmentation-based cropping to remove unnecessary information in the images with a class-inherent transformation network to obtain two generated transformed images from each input image and then used the transformations to diagnose the disease. The results we obtained in this study together with the first public quality data set support the need for such studies, especially in 2020.

Regarding the third and last objective, we proposed the RAFNI algorithm, which helps during the training process of the CNNs when there is some type of label noise in the training set. We have also made the code of the algorithm public, so it is possible to use it. The results we obtained in this study show that this algorithm is a good proposal that helps with any type of label noise and even can be used when there is no label noise, as it does not damage the learning in this case. This is important as the majority of times, we do not know if the data sets present label noise, which type or the noise rate.

8.2 FUTURE WORK

We can divide the future work in the same three branches.

For the classification of coral species, we think that it would be interesting to test the performance of some newer CNN architectures that have been developed since we performed this study that, while achieving state-of-the-art results on ImageNet, have fewer training weights and, as a result, are faster and easier to train with fewer data, like the EfficientNet family [TL19].

For the diagnosis of the COVID-19 disease, we want to release new versions of the data set that contain more images, fusing images from hospitals in other cities. The fusion of clinical information along with the images is another very interesting line of work. Finally, we also want to attempt the classification of the disease per severity, as a multi-class problem.

Finally, for the proposal to deal with label noise when using CNNs, we think that it would be interesting to further test the algorithm with real-world data sets that already include some type and rate of label noise, instead of introducing it by hand.

BIBLIOGRAPHY

[ARV+17] C. Affonso, A. L. D. Rossi, F. H. A. Vieira and A. C. P. de Leon Ferreira de Carvalho, 'Deep learning for biological image classification', *Expert Systems with Applications*, vol. 85, pp. 114–122, 2017.

[AHN+20] P. Afshar, S. Heidarian, F. Naderkhani, A. Oikonomou, K. N. Plataniotis and A. Mohammadi, 'Covid-caps: A capsule network-based framework for identification of covid-19 cases from x-ray images', *Pattern Recognition Letters*, vol. 138, pp. 638–643, 2020.

[ANB17] W. Alakwaa, M. Nassef and A. Badr, 'Lung cancer detection and classification with 3d convolutional neural network (3d-cnn)', *Lung Cancer*, vol. 8, no. 8, p. 409, 2017.

[AWA+19] E. Amid, M. K. Warmuth, R. Anil and T. Koren, 'Robust bi-tempered logistic loss based on bregman divergences', *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[AOA+19] E. Arazo, D. Ortego, P. Albert, N. O'Connor and K. McGuinness, 'Unsupervised label noise modeling and loss correction', in *International conference on machine learning*, PMLR, 2019, pp. 312–321.

[BEK+12] O. Beijbom, P. J. Edmunds, D. I. Kline, B. G. Mitchell and D. Kriegman, 'Automated annotation of coral reef survey images', in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 1170–1177.

[BMM18] M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', *Neural Networks*, vol. 106, pp. 249–259, 2018.

[Chu20] A. Chung, 'Figure 1 COVID-19 chest X-ray dataset initiative', 2020.

[CMD20] J. P. Cohen, P. Morrison and L. Dao, 'Covid-19 image data collection', *arXiv preprint arXiv:2003.11597*, 2020.

[CCD08] P. Cunningham, M. Cord and S. J. Delany, 'Supervised learning', in *Machine learning techniques for multimedia*, Springer, 2008, pp. 21–49.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'Imagenet: A large-scale hierarchical image database', in *Computer Vision and Pattern Recognition (CVPR), 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[Den12] L. Deng, 'The mnist database of handwritten digit images for machine learning research [best of the web]', *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[DWF+18] Y. Ding, L. Wang, D. Fan and B. Gong, 'A semi-supervised two-stage approach to learning from noisy labels', in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1215–1224.

[Ela15] M. Elawady, 'Sparse coral classification using deep convolutional neural networks', *arXiv preprint arXiv:1511.09067*, 2015.

[FAA+17] A. Farooq, S. Anwar, M. Awais and S. Rehman, 'A deep cnn based multi-class classification of alzheimer's disease using mri', in *2017 IEEE International Conference on Imaging systems and techniques (IST)*, IEEE, 2017, pp. 1–6.

[FP11] D. Forsyth and J. Ponce, *Computer vision: A modern approach*. Prentice hall, 2011.

[FV13] B. Frénay and M. Verleysen, 'Classification in the presence of label noise: A survey', *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[GWL+18] F. Gao, T. Wu, J. Li, B. Zheng, L. Ruan, D. Shang and B. Patel, 'Sd-cnn: A shallow-deep cnn for improved breast cancer diagnosis', *Computerized Medical Imaging and Graphics*, vol. 70, pp. 53–62, 2018.

[GLH15] S. García, J. Luengo and F. Herrera, *Data preprocessing in data mining*. Springer, 2015, vol. 72.

[GRL+16] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez and F. Herrera, 'Big data preprocessing: Methods and prospects', *Big Data Analytics*, vol. 1, no. 1, pp. 1–22, 2016.

[GKS17] A. Ghosh, H. Kumar and P. Sastry, 'Robust loss functions under label noise for deep neural networks', in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[GCM21] R. G. González-Acuña, H. A. Chaparro-Romo and I. Melendez-Montoya, *Optics and Artificial Vision*. IOP Publishing, 2021.

[GBC16] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. MIT press, 2016.

[HPK11] J. Han, J. Pei and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[HTF09] T. Hastie, R. Tibshirani and J. Friedman, 'Unsupervised learning', in *The elements of statistical learning*, Springer, 2009, pp. 485–585.

[HZR+16] 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.

[HMW+18] D. Hendrycks, M. Mazeika, D. Wilson and K. Gimpel, 'Using trusted data to train deep networks on labels corrupted by severe noise', *Advances in neural information processing systems*, vol. 31, 2018.

[HLV+17] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, 'Densely connected convolutional networks', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[IS15] S. Ioffe and C. Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[JNC16] I. Jindal, M. Nokleby and X. Chen, 'Learning deep networks from noisy labels with dropout regularization', in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, IEEE, 2016, pp. 967–972.

[KDR+20] M. Karim, T. Döhmen, D. Rebholz-Schuhmann, S. Decker, M. Cochez, O. Beyan *et al.*, 'Deepcovidexplainer: Explainable covid-19 predictions based on chest x-ray images', *arXiv preprint arXiv:2004.04582*, 2020.

[LBD+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, 'Backpropagation applied to handwritten zip code recognition', *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[LTJ21] J. Lu, L. Tan and H. Jiang, 'Review on convolutional neural network (cnn) applied to plant leaf disease classification', *Agriculture*, vol. 11, no. 8, p. 707, 2021.

[LGH12] J. Luengo, S. García and F. Herrera, 'On the choice of the best imputation methods for missing values considering three groups of classification methods', *Knowledge and information systems*, vol. 32, no. 1, pp. 77–108, 2012.

[MWH+18] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema and J. Bailey, 'Dimensionality-driven learning with noisy labels', in *International Conference on Machine Learning*, PMLR, 2018, pp. 3355–3364.

[MN20] G. Maguolo and L. Nanni, 'A critic evaluation of methods for covid-19 automatic detection from x-ray images', *arXiv preprint arXiv:2004.12823*, 2020.

[MBA+16a] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. Fisher, 'Automatic annotation of coral reefs using deep learning', in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–5.

[MBA+16b] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. Fisher, 'Coral classification with hybrid feature representations', in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 519–523.

[MAP+15] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[Min20] E. Mineo, *U-Net lung segmentation*, Accesible en: https://www.kaggle.com/eduardomineo/u-net-lung-segmentation-montgomery-shenzhen, 2020.

[NKP21] A. Narin, C. Kaya and Z. Pamuk, 'Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks', *Pattern Analysis and Applications*, vol. 24, no. 3, pp. 1207–1220, 2021.

[NMN+19] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel and T. Brox, 'Self: Learning to filter noisy labels with self-ensembling', *arXiv preprint arXiv:1910.01842*, 2019.

[NNL+19] D. T. Nguyen, T.-P.-N. Ngo, Z. Lou, M. Klar, L. Beggel and T. Brox, 'Robust learning under label noise with iterative noise-filtering', *arXiv preprint arXiv:1906.00216*, 2019.

[NZ08] M.-E. Nilsback and A. Zisserman, 'Automated flower classification over a large number of classes', in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, IEEE, 2008, pp. 722–729.

[OTY+20] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim and U. R. Acharya, 'Automated detection of covid-19 cases using deep neural networks with x-ray images', *Computers in biology and medicine*, vol. 121, p. 103 792, 2020.

[PGM+19] A. Paszke *et al.*, 'Pytorch: An imperative style, high-performance deep learning library', in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[PRK+17] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock and L. Qu, 'Making deep neural networks robust to label noise: A loss correction approach', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1944–1952.

[PF91] G. Piateski and W. Frawley, *Knowledge Discovery in Databases*. Cambridge, MA, USA: MIT Press, 1991.

[PRJ+08] O. Pizarro, P. Rigby, M. Johnson-Roberson, S. B. Williams and J. Colquhoun, 'Towards image-based marine habitat classification', in *OCEANS 2008*, IEEE, 2008, pp. 1–7.

[RSNA19] *Radiological society of north america. RSNA pneumonia detection challenge*, 2019.

[RGT+20] M. Rey-Area, E. Guirado, S. Tabik and J. Ruiz-Hidalgo, 'Fucitnet: Improving the generalization of deep learning networks by the fusion of learned class-inherent transformations', *Information Fusion*, vol. 63, pp. 188–195, 2020.

[RDG+95] D. E. Rumelhart, R. Durbin, R. Golden and Y. Chauvin, 'Backpropagation: The basic theory', *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[RDS+15] O. Russakovsky *et al.*, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[SGG+13] A. Shihavuddin, N. Gracias, R. Garcia, A. C. Gleason and B. Gintert, 'Image-based coral reef classification and thematic mapping', *Remote Sensing*, vol. 5, no. 4, pp. 1809–1841, 2013.

[Shi17] A. Shihavuddin, *Coral reef dataset, v2*. Mendeley data https://data.mendeley.com/datasets/86y667257h/2, Accessed on 12-02-2018, 2017.

[SZ14] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556*, 2014.

[SC18] G. Song and W. Chai, 'Collaborative learning for deep neural networks', *Advances in neural information processing systems*, vol. 31, 2018.

[SKL19] H. Song, M. Kim and J.-G. Lee, 'Selfie: Refurbishing unclean samples for robust deep learning', in *International Conference on Machine Learning*, PMLR, 2019, pp. 5907–5915.

[SH19] J. Speth and E. M. Hand, 'Automated label noise identification for facial attribute recognition.', in *CVPR Workshops*, 2019, pp. 25–28.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting', *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[SBP+14] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev and R. Fergus, 'Training convolutional networks with noisy labels', *arXiv preprint arXiv:1406.2080*, 2014.

[SLJ+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, 'Going deeper with convolutions', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[SVI+16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, 'Rethinking the inception architecture for computer vision', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[Sze10] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[TL19] M. Tan and Q. Le, 'Efficientnet: Rethinking model scaling for convolutional neural networks', in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[WW20] L. Wang and A. Wong, 'COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest radiography images', *arXiv preprint arXiv:2003.09871*, 2020.

[WKH+19] X. Wang, E. Kodirov, Y. Hua and N. M. Robertson, 'Improving mae against cce under label noise', *arXiv preprint arXiv:1903.12141*, 2019.

[WMC+19] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi and J. Bailey, 'Symmetric cross entropy for robust learning with noisy labels', in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 322–330.

[XXY+15] T. Xiao, T. Xia, Y. Yang, C. Huang and X. Wang, 'Learning from massive noisy labeled data for image classification', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

[YSH+19] Y. Yu, X. Si, C. Hu and J. Zhang, 'A review of recurrent neural networks: Lstm cells and network architectures', *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[ZBH+21] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, 'Understanding deep learning (still) requires rethinking generalization', *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[ZLZ+18] Q.-x. Zhang, G.-h. Lin, Y.-m. Zhang, G. Xu and J.-j. Wang, 'Wildland forest fire smoke detection based on faster r-cnn using synthetic smoke images', *Procedia engineering*, vol. 211, pp. 441–446, 2018.

[ZS18] Z. Zhang and M. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', *Advances in neural information processing systems*, vol. 31, 2018.

[ZG09] X. Zhu and A. B. Goldberg, 'Introduction to semi-supervised learning', *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

Part II

PUBLICATIONS

9

TOWARDS HIGHLY ACCURATE CORAL TEXTURE IMAGES CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS AND DATA AUGMENTATION

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A. S. M., Krawczyk, B., & Herrera, F. (2019). Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation. *Expert Systems with Applications*, 118, 315-328.
DOI: https://doi.org/10.1016/j.eswa.2018.10.010

- Status: Published
- Impact Factor (JCR 2019): 5.452
- Subject Category: Computer Science, Artificial Intelligence. Ranking 21/137 (Q1)
- Subject Category: Engineering, Electrical & Electronic. Ranking 32/266 (Q1)
- Subject Category: Operations Research & Management Science. Ranking 2/83 (Q1)

TOWARDS HIGHLY ACCURATE CORAL TEXTURE IMAGES CLASSIFICATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS AND DATA AUGMENTATION

Anabel Gómez-Ríos[a], Siham Tabik[a], Julián Luengo[a], ASM Shihavuddin[b], Bartosz Krawczyk[c], Francisco Herrera[a]

[a] Andalusian Research Institute in Data Science and Computational Intelligence, Dept. of Computer Science and AI, University of Granada, Granada, Spain
[b] Dept. of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Kgs. Lyngby, Denmark
[c] Dept. of Computer Science, Virginia Commonwealth University, USA

ABSTRACT

The recognition of coral species based on underwater texture images poses a significant difficulty for machine learning algorithms, due to the three following challenges embedded in the nature of this data: 1) datasets do not include information about the global structure of the coral; 2) several species of coral have very similar characteristics; and 3) defining the spatial borders between classes is difficult as many corals tend to appear together in groups. For this reasons, the classification of coral species has always required an aid from a domain expert. The objective of this paper is to develop an accurate classification model for coral texture images. Current datasets contain a large number of imbalanced classes, while the images are subject to inter-class variation. We have focused on the current small datasets and analyzed 1) several Convolutional Neural Network (CNN) architectures, 2) data augmentation techniques and 3) transfer learning approaches. We have achieved the state-of-the art accuracies using different variations of ResNet on the two small coral texture datasets, EILAT and RSMAS.

Keywords: *Coral Images Classification, Deep Learning, Convolutional Neural Networks, Inception, ResNet, DenseNet*.

9.1 INTRODUCTION

Coral reefs are complex marine ecosystems typical to the warm and shallow seas of the tropics. The reefs are created by the slow accumulation of hard calcium carbonate skeletons that hard coral species leave behind when they die, waiting for another coral to live in it and expand the reef. Coral reefs are one of the most valuable ecosystems in the world as they are extremely biodiverse. They support up to two million species and a quarter of all marine life on Earth [ESI17]. They are also very important from the human point of view [FBS+14]. Coral species help to clean the water and remove nitrogen and carbon, they are a source for medicine research and economic wealth from fishing and tourism, they are also a natural barrier for coastal protection against hurricanes and storms and, since many of them are thousands and even millions years old, their study helps scientists to understand climatic events of the past.

The study of the distribution of coral reefs over time can provide important clues about the impact of global warming and water pollution levels. According to *Endangered Species International* [ESI17], we have already lost 19% coral reefs areas since the 1950s and, according to the International Union for Conservation of Nature (IUCN) Red List of Threatened Species [IUCN17], in 2017 there were 237 threatened species in the evaluated 40% of the estimated total of species. This is due to the facts that coral reefs do not tolerate temperature changes and a quarter of the carbon dioxide emissions in the atmosphere is absorbed by the ocean, in addition to the water pollution and other problems caused by humans. An extensive study on coral reef extension loss and growth can be found in Pratchett *et al.* [PAH+15].

With recent advancements in image acquisition technologies and growing interest in this topic among the scientific community, huge amount of data on coral reefs is being collected. However, it is complicated to keep a record of all coral species because there are thousands of them and the taxonomy is mutable. This is due to new discoveries made by scientists or because they may change the order, family or genus of existing species as they gather more knowledge about them. In addition, some coral species have different sizes, shapes and colors, but other coral species seem to be identical for a human observer. As a consequence, a successful coral classification has always demanded an expert biologist. If we can automate the classification by using the amount of coral images that is being collected, we can help scientists to study more closely that amount of data, making an important step towards automatic knowledge discovery process. In fact, automatizing the classification process of coral images has been addressed in a few number of works. Most of them [BEK+12; PRJ+08;

SGG+13; SD09] use machine learning models combined, in some cases, with image enhancement techniques and feature extractors. Ani Brown Mary and Dharma [AD17] and Ani Brown Mary and Dejey [AD18b; AD18a] proposed new feature extractors based on Local Binary Pattern (LBP). Among these works, only Shihavuddin *et al.* [SGG+13], Ani Brown Mary and Dharma [AD17] and Ani Brown Mary and Dejey [AD18b; AD18a] use several datasets.

In recent years, Convolutional Neural Networks (CNNs) have shown outstanding accuracies for image classification [KSH12; RDS+15], especially in the field of Computer Vision. Currently, their applications branch out to a plethora of diverse fields, where analysis of image data is required. In biology, CNNs have been evaluated and compared with machine learning algorithms for wood classification [ARV+17]. In coral classification, the use of CNNs is challenging due to the variance between images of the same class, the lightning variations due to the water column or the fact that some coral species tend to appear together. Besides, CNNs need a large dataset to achieve a good performance. In practice, two techniques are used to overcome this limitation: transfer learning and data augmentation. There are some works that use CNNs for coral classification [Ela15; MBA+16a; MBA+16b; AD18a], but they evaluate popular CNNs, like VGGnet or LeNet and they only use one dataset to test their models. Besides, they do not analyze EILAT or RSMAS, which are very interesting datasets due to their small sizes, they are highly imbalanced and they include only small parts of the corals texture, the images do not include any information on the entire body of the corals.

We propose to use more capable CNNs to overcome the limitations of previously applied deep learning models. We want to develop a much more accurate model approaching the human expert, facing the specific problems of coral classification using several datasets. In particular, we have considered three of the most promising CNNs, Inception v3 [SVI+16], ResNet [HZR+16] and DenseNet [HLV+17]. Inception is a newer version of GoogleNet [SLJ+15], which won the ImageNet Large Scale Visual Recognition Competition (ISLVRC) [RDS+15] in 2014. ResNet won the same competition in 2015 and DenseNet beat the results of ResNet in 2016. For the classification, we have considered two underwater coral datasets, RSMAS and EILAT. To evaluate different ways of transfer learning, we have also used a larger domain dataset, the MLC-2008 dataset [BEK+12], which includes more than 43,000 images distributed into 9 classes, 5 coral classes and 4 non-coral classes. We will use MLC-2008 for fine-tuning the networks weights as a previous step of RSMAS and EILAT

classification. We have compared our results with the current most accurate model [SGG+13].

The contributions of this work are the following:

- Study, explore and analyze the performance of the most promising CNNs in the classification of small datasets of underwater coral texture images.
- Analyze the impact of transfer learning from ImageNet versus transfer learning from a coral domain dataset, MLC-2008, on the classification of small coral texture datasets.
- Analyze the impact of data augmentation on the performance of the coral texture classification model.
- Compare our results with the state-of-the-art classical methods which require high human supervision and intervention.

The rest of the paper is organized as follows. An overview of the three considered CNNs is provided in Section 9.2. The challenges of coral classification and related works are given in Section 10.3. A description of the coral datasets we have used is provided in Section 10.4. A description of the experimental framework we have used in all the experiments we have carried out in this paper is provided in Section 9.5. The experiments and results are given in Section 12.5 and the final conclusions of this study are given in Section 12.9.

9.2 CNN CLASSIFICATION MODELS

CNNs have achieved outstanding accuracies in a plethora of contemporary applications, automatizing its design [FCN+18]. In fact, since 2012 the prestigious ILSVRC competition [RDS+15] has been won exclusively by CNN models. The CNN layers capture increasingly complex features as the depth increases. In recent years, these architectures have evolved by increasing first the depth of the networks, then the width and finally using lower features obtained from the lower layers into higher layers. This section provides an overview of the CNNs used in this work. We have considered three influential CNNs, Inception v3 (Subsection 9.2.1), ResNet (Subsection 9.2.2), and one of the newest, DenseNet, (Subsection 9.2.3). Finally, we describe the optimization techniques that we have used to overcome the small sizes of the considered datasets (Subsection 9.2.4).

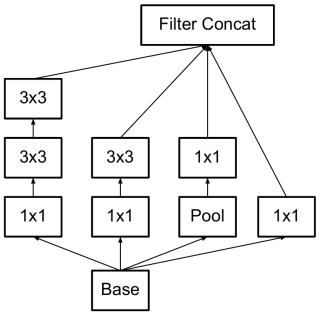


Figure 9.1: Base Inception v3 module. Figure from [SVI+16].

9.2.1 *Inception v3*

GoogLeNet [SLJ+15] won the ILSVRC in 2014 and it is based on the repetition of a module called inception. This module have six convolutions and one max-pooling. Four of these convolutions use a 1×1 kernel, which is introduced to increase the width and the depth of the network and to reduce the dimensionality when it is necessary. In this sense, a 1×1 convolution is performed before the other two convolutions in the module, a 3×3 and a 5×5 convolution. After all the computation, the output of the module is calculated as the concatenation of the output of the convolutions. This module is repeated 9 times and at the end it uses a dropout layer. In total, GoogLeNet has 22 learnable layers.

Inception v3 can be considered as a modification of GoogLeNet. The base inception module is changed by removing the 5×5 convolution and introducing instead two 3×3 convolutions, as we can see in Figure 9.1. The resulting network is made up of 10 inception modules. Furthermore, the base module is modified as the network goes deeper. Five modules are changed by replacing the $n \times n$ convolutions by a 1×7 followed by a 7×1 convolution in order to reduce the computational cost. The last two modules replace the last two 3×3 convolutions by a 1×3 and a 3×1 convolutions each one, this time in parallel. Lastly, the first 7×7 convolution in GoogLeNet is also changed by three 3×3 convolutions. In total, Inception v3 has 42 learnable layers.

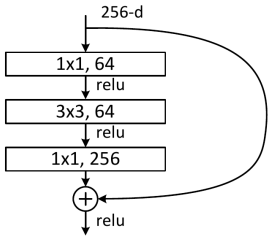


Figure 9.2: ResNet building block. Figure from [HZR+16].

9.2.2 *ResNet*

Increasing the network's depth to obtain a better precision makes the network more difficult to optimize since it may produce the vanishing or exploding gradients problem. ResNet [HZR+16], which won the ILSVRC classification task in 2015, address this issue by fitting a residual mapping instead of the original mapping, and by adding several connections between layers. These new connections skip various layers and perform an identity, which not adds any new parameters, or a simple 1×1 convolution. In particular, this network is also based on the reiterated use of a module, called a building block. The depth of the network depends on the number of the used building blocks. For 50 or more layers, the building block consists of three convolutions, a 1×1 followed by a 3×3 followed by a 1×1 convolution, and a connection joining the input of the first convolution to the output of the third convolution, as we can see in Figure 9.2. For our problem, we have used the model with 50 layers, ResNet-50, and with 152 layers, ResNet-152.

9.2.3 *DenseNet*

DenseNet is also based on the repetition of a block, called the dense block. Inspired by the building block of ResNet, DenseNet connects the output of all the layers to the input of all the following layers within the dense block [HLV+17]. The connections between blocks, called transition layers, work as a compression factor in a sense that the transition layer generates less feature maps than it receives. The difference between connections in the dense block and connections in the building block of ResNet is that in the dense block the outputs of previous layers are added to the following layers before its computation is performed. A dense block is the repetition

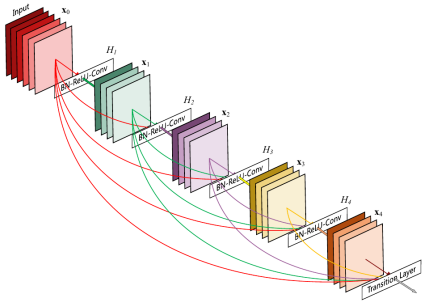


Figure 9.3: Example of a dense block. Figure from [HLV+17].

of a Batch Normalization, a ReLU, a 1×1 convolution, a Batch Normalization, a ReLU and a 3×3 convolution a specific number of times. In Figure 9.3 we can see an example of a DenseNet block. The transition layers are a 1×1 convolution followed by an average pooling with kernel 2×2. Similarly to ResNet, the number of dense blocks determines the number of layers in the network. In this work, we have analyze DenseNet-121 and DenseNet-161, which include 121 and 161 layers respectively.

9.2.4 *CNN Optimization Techniques*

CNN-based models require a large set of training samples to achieve good generalization capabilities. However, generating large datasets is either costly, time-consuming, or sometimes simply impossible. In practice, two techniques are used to overcome this limitation: transfer learning and data augmentation. Since the current coral datasets are too small to train an effective CNN from scratch, we propose to use these two approaches:

- **Transfer learning**: instead of starting the training from scratch by randomly initializing the weights, we initialize the weights using a pre-trained network on a different dataset, usually much larger in size. In this work, we have considered using the knowledge learned from the massive common objects dataset ImageNet [DDS+09] and from an expert domain dataset, MLC-2008 [BEK+12]. Since EILAT and RSMAS are too small to fine-tune all the weights of the pre-trained networks, we have evaluated two alternatives. First, we have used the pre-trained networks on ImageNet, we have added two fully connected layers and we have trained these two fully connected layers and freeze all the

previous layers to classify RSMAS and EILAT. Second, we have used the pre-trained networks on ImageNet, fine-tuned all the weights with MLC-2008 and then we have added the same two fully connected layers at the end and trained them on RSMAS and EILAT, freezing all the previous layers.

- **Data augmentation**: consists of artificially increasing the volume of the training set by applying several distortions to the original images, such as changing the brightness, scaling or zooming, rotation, vertical or horizontal mirroring, etc. The applied distortions should not alter the spatial pattern of target classes. Usually the distortions are performed during the training time, which allows to do it on the fly without saving the new images.

9.3 PREVIOUS ADVANCES ON AUTOMATIC CORAL REEF CLASSIFICATION

In this section we explain the challenges of the underwater coral and coral reef images classification and we give an overview on existing works for automating the classification of coral reef habitat using underwater imagery. The reasons why the classification of such images is difficult are provided in Subsection 9.3.1. Previous works in coral classification can be divided into two groups, methods that combine classical models (Subsection 9.3.2) and methods that use CNNs (Subsection 9.3.3).

9.3.1 *Challenges of Coral Classification*

The classification of underwater coral images is challenging for the following reasons:

- Partial occlusion of objects due to the three-dimensional structure of the seabed. Depending on the water type, there can be significant variation in presence of scattering effect, which increases additive noise on the image acquisition and makes it difficult for any computer vision algorithm to perform as in a normal environment.
- Lightning variations due to wave focusing and variable optical properties of the water column. In the deep underwater scenario, it is common that there is no natural light source other than the remote sensing device, which implies non uniform illumination across the acquired images.
- Subjective annotation of the training samples by different analysts.

- Variation in viewpoints, distances and image quality.
- Significant inner-class variability in the morphology of benthic organisms.
- Complex spatial borders between classes, as many coral species tend to appear together.
- There are very few datasets of underwater coral reef images and in general they contain patches of the texture of the corals, while at the same time they do not include any information on the global structure of the coral.

9.3.2 *Coral Classification Based on Classical Methods*

Most of the existing approaches for classifying underwater coral images combine one feature extractor with a classifier and show their performance only using a single dataset, i.e., with specific size, resolution of the images, number of classes and color information [BEK+12; PRJ+08; SD09]. The first paper in this subject was by Pizarro *et al.* [PRJ+08]. The authors analyzed more marine habitat besides corals, so it is more general. They used a SIFT descriptor and a bag of features approach, which means that they chose from the training set the images that are more similar to each test image. Beijbom *et al.* [BEK+12] introduced the Moorea Labelled Corals (MLC) dataset, which has large images containing different coral species, and they used Support Vector Machines along with filters and a texture descriptor. They obtained an accuracy of 83.1% on this dataset using the images of 2008 and 2009 for training and the images of 2010 for testing. Stokes and Deane [SD09] used a normalized color space and a discrete cosine transform to extract texture features. Again, they only used one dataset, provided by the National Oceanic and Atmospheric Administration (NOAA) of the U.S. Department of Commerce Ocean Explorer.

[SGG+13] developed an unified classification algorithm for four different datasets of different characteristics, in which we can find RSMAS and EILAT. The authors combined multiple image enhancement techniques, feature extractors and classifiers, among other techniques. In particular, the image enhancement step contain four algorithms, one mandatory (Contract Limited Adaptive Histogram Specification or CLAHS) and three optional (color correction, normalization and color channel stretching). The feature extraction step contain one optional, used as color descriptor, and three mandatory algorithms, used as texture descriptors. Then, the method has a kernel mapping step with three mandatory algorithms, a dimension reduction

step with two optional algorithms and a prior settings step with one algorithm. Then it performs the classification using one of the following algorithms: multiclass SVM, KNN, a neural network or probability density weighted mean distance. Lastly, if the original image was a mosaic containing several patches, it uses a thematic mapping using sliding window and morphological filtering. By configuring the hyperparameters and the different combinations of these algorithms, the model can be adapted to different datasets.

This method is considered to be the state-of-the-art for RSMAS and EILAT datasets. In particular, in these two datasets the best combination of algorithms is the following: in the image enhancement step it uses just CLASH. In the feature extraction step it uses the opponent angle histogram, the hue channel histogram, the gray level co-occurrence matrix, the Completed Local Binary Pattern (CLBP) and the Gabor filter response. In the kernel mapping step it uses L1 normalization, chi-square kernel and Hellinger kernel for CLBP and the color histogram. In the dimension reduction step it uses principal component analysis and Fisher kernel. In the prior settings step it uses class frequency to estimate prior probability. Finally, as classification algorithm it uses KNN.

As it can be seen, this algorithm implies a lot of human supervision and intervention, as a lot of algorithms had to be evaluated with several hyperparameters and there were many possible combinations between them. Furthermore, when we have the best combination we need to use a lot of algorithms every time we need to classify a new image. In the particular case of EILAT and RSMAS, we need to use six algorithms to obtain the classifier and every time we need to classify a new image we need to use the first four algorithms, until we obtain the features of the image.

Ani Brown Mary and Dharma [AD17] developed an improved Local Binary Pattern (LBP) called ILBP which obtained diagonal pattern features in the images. To test their method, they used several datasets, including EILAT, EILAT2 (a subset of EILAT), RSMAS and a subset of MLC-2012. They reported very good accuracies on these datasets.

Ani Brown Mary and Dejey [AD18b] proposed another modification of the LBP feature descriptor called Z with Tilted Z LBP which reduced the computational complexity of LBP. The coral images were enhanced using Contrast Limited Adaptative Histogram Equalization. They used several datasets, including one coral video. Among others, they used EILAT, EILAT2, RSMAS and MLC-2012. The results with this feature extractor were in general better, except for EILAT2, than the ones

obtained in [AD17]. We will compare our results with the ones obtained in these two works.

Shakoor and Boostani [SB18] developed an advanced LBP for the classification of coral datasets. In particular, they proposed two mapping methods and a new combination of LBP to extract the features. They tested their methods with several datasets, between which we can find EILAT2, a subset of RSMAS with 8 classes and a subset of MLC-2008 with 2055 images. However, the obtained accuracies were considerably lower than the ones obtained by Shihavuddin *et al.* [SGG+13].

9.3.3 *Coral Classification Based on CNNs Methods*

The use of CNNs for coral classification allow us to use the images without the image enhancements, although it is possible to use them, and without the feature extraction, saving a lot of experiments to detect the best combination of algorithms and therefore, saving time.

The first work that used CNNs for coral classification was by Elawady [Ela15]. The author first enhanced the input raw images via color correction and smoothing filtering. Then, he trained a LeNet-5 [LBB+98] based model whose input layer consisted of three basic channels of color image plus extra channels for texture and shape descriptors consisting of the following components: zero component analysis whitening, phase congruency, and Weber local descriptor. The model obtained around 55% accuracy on the two used datasets.

Mahmood *et al.* [MBA+16a] used VGGnet [SZ14] pretrained on ImageNet and the dataset BENTHOZ-2015 [BFF+15] to fine-tune the network. This dataset contains more than 400,000 images and associated sensor data collected by an autonomous vehicle over Australia. The authors extracted several patches from each image centered in different pixels and using different scales and they applied a color channel stretch to the patches as a pre-processing technique. In this article, they proposed a mechanism to automatically label unseen coral reef images to obtain the coral coverage in the region where the images were collected (i.e., classifying new images as coral or non-coral ones). A marine expert later verified the accuracy of this automatic method. In the presented experimental study, authors conducted several experiments and reported over 90% accuracy obtained in each of them.

Mahmood *et al.* [MBA+16b] used the MLC dataset to propose the usage of CNNs along with hand-crafted features. Moreover, they introduced a mechanism to extract

such features. This proposal is based on the observation that CNNs cannot be trained from scratch using the available coral datasets due to its small size. The features extraction with CNNs was carried out with the network VGGnet pre-trained on ImageNet. To classify both types of features, they used a two layer Perceptron. In their experiments, they obtained better accuracies with this technique than just with VGGnet or just the hand-crafted features, although the difference between VGGnet and the combination of the features is small. In their best experiment, they obtained an accuracy of 84.5% in MLC.

Beijbom *et al.* [BTK+16] proposed the use of fluorescence images along with usual images to improve the classification. They created a dataset collecting common images and fluorescence images in Eilat using a FluorIS system, which they had developed previously. They reported a 22% reduction of classification error compared to use only common images.

Ani Brown Mary and Dejey [AD18a] proposed a new feature descriptor called Octa-angled Pattern for Triangular sub region (OPT) and its use along with a proposed CNN called Pulse Coupled Convolutional Neural Network (PCCNN). The feature descriptor used diagonal and center elements of the neighborhood of a pixel to obtain the features and the images were not enhanced before this step. The PCCNN was used to reduce the number of features generated by CNNs. To test their method, the authors used several coral datasets, including EILAT, EILAT2, RSMAS and MLC-2012. This method reported outstanding results for these datasets, so we will compare our results with these ones.

These works use classical CNNs: VGGnet and LeNet, and they do not use EILAT or RSMAS. Besides, sometimes the accuracies obtained are low [Ela15], the classification is simple [MBA+16a], they use hand-made feature extraction along with CNNs [MBA+16b; AD18a] or fluorescence images [BTK+16] are not available.

9.4 DATASETS

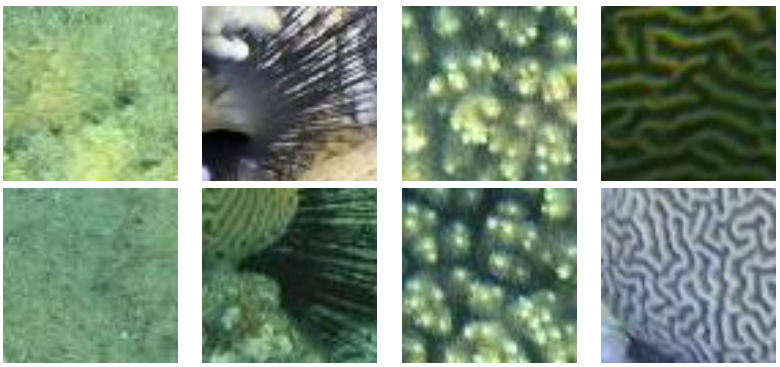
There exist eight open benchmarks for underwater coral classification. These include five public color datasets: EILAT, RSMAS, MLC, EILAT2 (a subset of EILAT) and the Red Sea Mosaic dataset. The remaining three are black and white datasets: UIUCtex, CURET and KTH-TIPS. Some of them include non-coral classes such as fabric, wood and brick. It is worth to mention that the Red Sea Mosaic dataset is actually one single large image that contains a large number of different coral species.

In this work, we have used the most recent and smallest RGB datasets that contain the highest number of coral species, RSMAS and EILAT [Shi17]. These two datasets are comprised of patches of coral images. These patches capture mainly the texture of different parts of the coral and do not include any information of the global structure of the entire coral. The usage of CNNs with texture images has already been successfully carried out for granite tiles classification [FG17]. In this case, both datasets are small with a large number of classes and also imbalanced. Some classes have a high number of samples whereas other classes include very few samples, which makes the classification more difficult. To evaluate the impact of transfer learning from a related domain dataset, we have also used the MLC-2008 dataset [BEK+12]. The main characteristics of these three datasets are as follows:

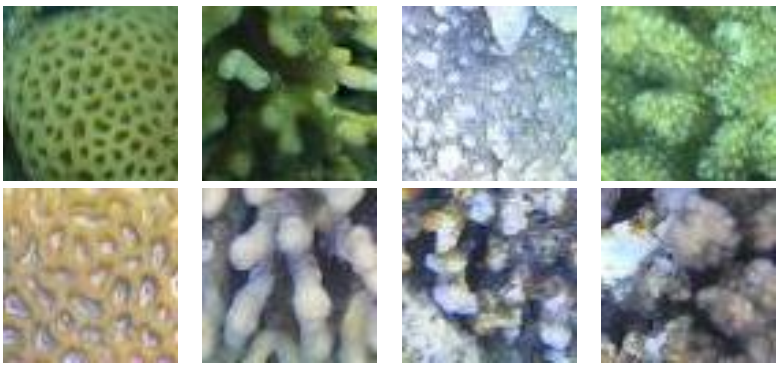
- EILAT contains 1123 image patches of size 64×64, taken from coral reefs near Eilat in the Red sea. The image patches are pieces of larger images. The original images were taken under equal conditions and with the same camera. See examples of patches in Figure 9.4. The patches have been classified into eight classes, but the used labels do not correspond to the coral species names. EILAT is characterized by imbalanced distribution of examples among classes, as it can be seen in Table 10.3.
- RSMAS contains 766 image patches of size 256×256. The images were collected by divers from the Rosenstiel School of Marine and Atmospheric Sciences of the University of Miami. These images were taken under different conditions as they were taken with different cameras in different places. See examples in Figure 10.3. The patches have been classified into 14 classes, whose labels correspond to the names of the coral species in Latin, as it can be seen in Table 10.3.
- MLC-2008 contains 43832 image patches of size 312×312 distributed into 9 classes, 5 coral classes and 4 non-coral classes. The coral classes are not the same as the classes used neither in RSMAS nor in EILAT.

9.5 EXPERIMENTAL FRAMEWORK

In this section we describe the experimental framework we have used to analyze and compare the considered models. First, we describe the software and hardware



(a) Sand (b) Urchin (c) BranchesI (d) Brain Coral



(e) Favid Coral (f) BranchesII (g) Dead Coral (h) BranchesIII

Figure 9.4: Selected patches from EILAT. Each column shows two examples per class.

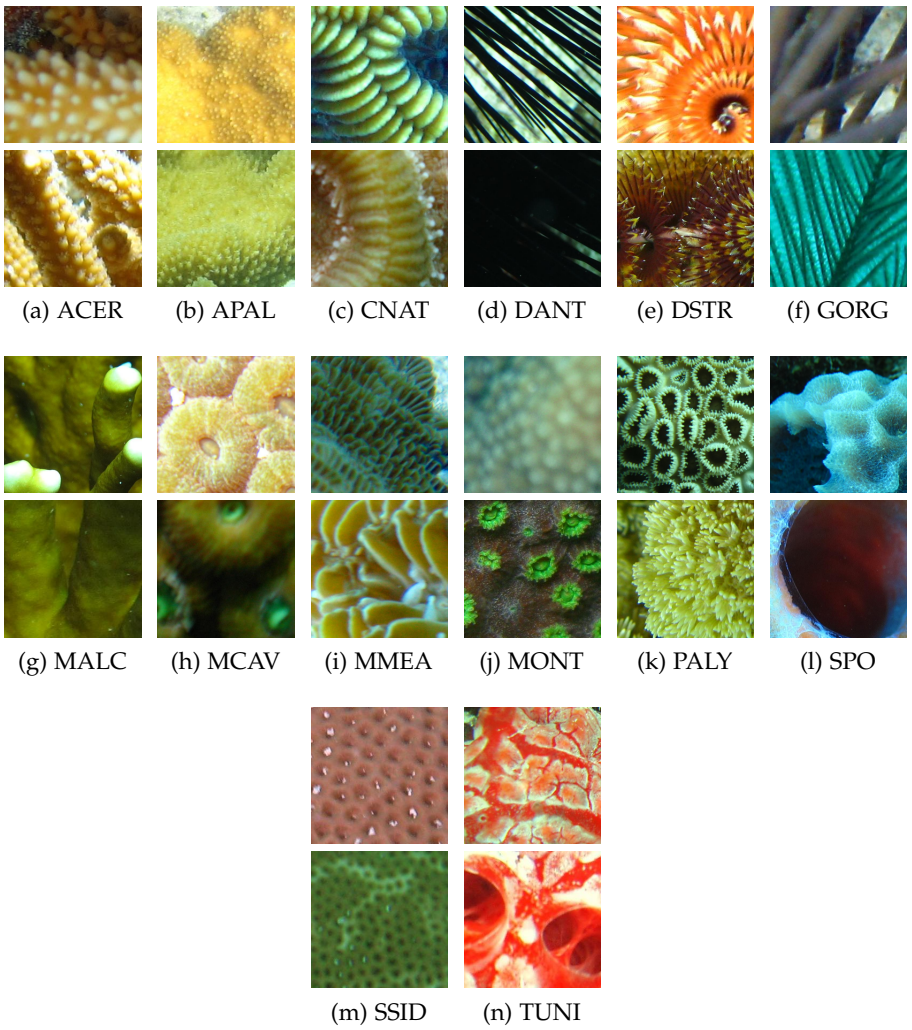


Figure 9.5: Selected patches from RSMAS. Each column shows two examples per class.

Table 9.1: Characteristics of EILAT and RSMAS. The #imgs refers to the number of images in the corresponding class.

Dataset	Classes	#imgs
EILAT	Sand.	87
	Urchin.	78
	Branches Type I.	29
	Brain Coral.	160
	Favid Coral.	200
	Branches Type II.	216
	Dead Coral.	296
	Branches Type III.	11
RSMAS	Acropora Cervicornis (ACER).	109
	Acropora Palmata (APAL).	77
	Colpophyllia Natans (CNAT).	57
	Diadema Antillarum (DANT).	63
	Diploria Strigosa (DSTR).	24
	Gorgonians (GORG).	60
	Millepora Alcicornis (MALC).	22
	Montastraea Cavernosa (MCAV).	79
	Meandrina Meandrites (MMEA).	54
	Montipora spp. (MONT).	28
	Palythoas Palythoa (PALY).	32
	Sponge Fungus (SPO).	88
	Siderastrea Siderea (SSID).	37
	Tunicates (TUNI).	36

we have used to evaluate the CNNs (Subsection 9.5.1) and the evaluation metric we have used to compare the results (Subsection 9.5.2). Then, we describe how we have performed transfer learning (Subsection 9.5.3) and the data augmentation techniques we have used (Subsection 9.5.4). Finally, we give all the hyperparameters we have evaluated in the CNN models (Subsection 9.5.5).

9.5.1 *Software and Hardware*

To evaluate Inception, ResNet and DenseNet, we have used Keras [Cho+15] as front-end and Tensorflow [MAP+15] as back-end. For Inception we have used the implementation available in Keras version 2.0.4 while for ResNet and DenseNet, we have adapted the code available in GitHub by Yu [Yu17].

All the CNN models have been evaluated on a NVIDIA Titan Xp, with 12GB of GDDR5X memory with 11.4 Gbps of frequency, and 3,840 cores with a frequency of 1.6 GHz.

9.5.2 *Evaluation Metric*

All the results provided in this paper have been obtained performing a 5 fold cross-validation technique. To analyze and compare the performance of different CNN architectures, configurations and optimizations, we have used the mean of the five accuracies obtained in the five folds. The accuracy is calculated as follows:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{N} \ ,$$

where N is the total number of instances.

9.5.3 *Transfer Learning*

We have used transfer learning by initializing the considered CNNs with the pre-trained weights of the networks on ImageNet. We have removed the last layer in each network, which is the layer that classifies the images into ImageNet classes. Afterwards, we have added to each model two fully connected layers, the first one with 512 neurons followed by a ReLU activation layer and the second fully connected layer with as many neurons as classes in the dataset followed by a softmax activation

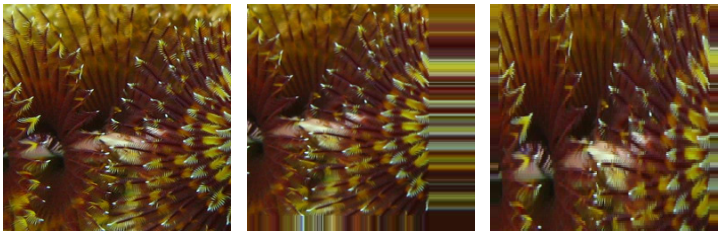
layer. That is, the last fully connected layer in each model has 8 neurons when classifying EILAT and 14 neurons when classifying RSMAS. As we are dealing with very small datasets, we have frozen all the layers except the last two fully connected layers, hence we only train the last two fully connected layers. We have used the Stochastic Gradient Descent as optimizer with a learning rate of 0.001, a decay of 10^{-6} and a Nesterov momentum of 0.9.

The process is similar when we have used fine-tuning with MLC-2008. We have used the pre-trained networks on ImageNet, we have removed the last layer in each network and added a fully connected layer with 9 neurons since MLC-2008 has 9 classes. As MLC-2008 is a large dataset, we have fine-tuned all the weights in the networks. When the training was completed, we have removed the layer with 9 neurons and added the same two fully connected layers as before: the first one with 512 neurons and ReLU activation and the second one with 8 neurons for EILAT and 14 neurons for RSMAS and softmax activation in both cases. Then we have frozen all the previous layers and we have only trained the last two fully connected layers.

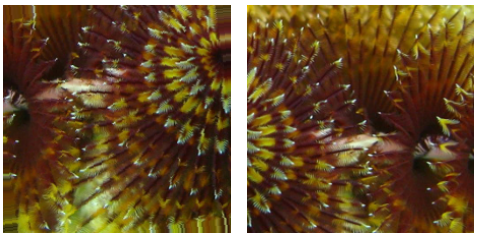
9.5.4 *Data Augmentation*

We have analyzed the impact of the following data augmentation techniques on the performance of the learning process:

- Random shift (referred to later as shift) consists of randomly shifting the images horizontally or vertically by a factor calculated as the fraction of the width or length of the image. In this work we shift the images horizontally and vertically in all the cases. Given a number x, the width and length of the image will be shifted by a random factor selected in the interval $[0, x]$.
- Random zoom (referred to later as zoom) consists of randomly zooming the image by a certain range. Given a value x, each image will be resized in the interval $[1 - x, 1 + x]$.
- Random rotation (referred to later as rotation) consist of randomly rotating the images by a certain angle. Given a value x, each image will be rotated by an angle in $[0, x]$.
- Random horizontal flip (referred to later as flip) consist of randomly flipping the images horizontally.



(a) Original (b) Shift (c) Zoom



(d) Rotation (e) Flìp

Figure 9.6: The result of applying four data augmentation techniques to (**a**) a original RSMAS image: (**b**) shift, (**c**) zoom, (**d**) rotation and (**e**) flip.

An illustration of these data augmentation techniques is shown in Figure 9.6. As it can be seen, the distorted images maintain the original size and the pixels outside the boundaries are filled with the values of the limit pixels. This effect can be clearly seen in Figure 9.6b.

9.5.5 *Hyperparameters*

Lastly, we have evaluated the impact of different hyperparameters on the performance of the analyzed networks, such as the number of layers, the number of epochs and the batch size. Specifically, we have used 50 and 152 layers for ResNet and 121 and 161 layers for DenseNet, so we have evaluated five CNNs models: Inception, ResNet-50, ResNet-152, DenseNet-121 and DenseNet-161. For each one of these CNNs, we have performed a grid search over the following parameters: batch size $= \{32, 64, 128\}$ and number of epochs $= \{100, 300, 500, 700, 1000, 1300\}$. To maintain a reasonable utilization of the GPU resources, we have considered the value 32 as the minimum batch size.

9.6 CLASSIFICATION OF CORAL TEXTURE IMAGES WITH CNNS

This section is organized in four parts. First, we analyze the results of our CNN-based classifiers without data augmentation. We analyze here transfer learning from ImageNet versus transfer learning from ImageNet and fine-tuning with MLC-2008. We also analyze the impact of class balancing based on a cost-sensitive loss function. Then, we compare our best results with the state-of-the-art models on EILAT and RSMAS (Subsection 9.6.1). Second, we analyze the impact of data augmentation on the two small coral texture datasets, EILAT and RSMAS, using the best approach found in Subsection 9.6.1 (Subsection 9.6.2). Third, we provide a deeper analysis on the missclassified EILAT and RSMAS images by their best models (Subsection **??**). Finally, we explore the use of the best approach found for EILAT and RSMAS in other datasets (Subsection 9.6.4).

9.6.1 *Classification of Coral Texture Images without Data Augmentation*

In this subsection we have evaluated exhaustively Inception v3, ResNet and DenseNet with different hyperparameters and we have compared the results obtained for these

Table 9.2: The accuracies obtained by Inception v3, ResNet-50, ResNet-152, DenseNet-121, DenseNet-161 and the classical state-of-the-art Shihavuddin model. The results of all the Convolutional Neural Networks (CNNs) were obtained without data augmentation. The best results are stressed in bold.

	Shihavuddin's method	Inception v3	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
EILAT	95.79	96.23	**97.85**	**97.85**	91.03	93.81
RSMAS	92.74	96.71	97.67	**97.95**	89.73	91.10

Table 9.3: The set of hyperparameters that provides the best performance shown in Table 9.2 for each CNN model and the time it took to complete the 5 cross-validation process.

		Shihavuddin's method	Inception v3	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
EILAT	Batch Size	—	32	64	64	32	32
	Epochs	—	700	500	300	300	700
	Time (min)	23.35	7.36	3.34	4.32	4.66	8.88
RSMAS	Batch Size	—	32	64	32	32	64
	Epochs	—	1300	1300	300	700	1000
	Time (min)	81.90	9.07	5.08	3.71	5.22	5.27

three CNNs with the state-of-the-art model by Shihavuddin *et al.* [SGG+13] on EILAT and RSMAS. For Inception, we have analyzed the impact of different numbers of iterations and batch sizes. For ResNet and DenseNet, we have evaluated different combinations of number of epochs, batch sizes and network depths. We have also analyzed the impact of fine-tuning all the weights in the networks using a related and larger dataset, MLC-2008, and the impact of addressing the class imbalance problem in EILAT and RSMAS.

Shihavuddin *et al.* [SGG+13] compare themselves with the methods used in previous works, including previous works on coral reef image classification [BEK+12; PRJ+08; SD09] and state-of-the-art texture classification algorithms. They re-implemented the methods to use the same datasets they used in their work. Among these works, Shihavuddin *et al.* [SGG+13] reported the best results, so it is enough to compare our results with the ones obtained in this work.

As the results provided by Shihavuddin *et al.* [SGG+13] were performed using a 10 fold cross validation, we have re-evaluated their model using a 5 fold cross validation with the same folds we have used for all other models in order to compare them under the same conditions. We have also used the best hyperparameters for each dataset at each step of the method described in Subsection 9.3.2.

Since Shihavuddin's work, there have been more works using EILAT and RSMAS [AD17; AD18b; AD18a; SB18]. Among these, Shakoor and Boostani [SB18] obtained worse accuracies RSMAS (they did not used EILAT) than Shihavuddin *et al.* [SGG+13]. The other three works reported very good results on these two datasets, and all three used the same experimental framework to obtain their results. They used only one held-out test set of 10%, 25% or 50% of the images in the datasets instead of using a cross validation technique. The cross validation technique gives more stable and reliable results than a held-out test, so we continue to use the 5 fold cross validation. That way, we are testing with 20% of the images in each fold, so we compare our results with their 10% and 25% held-out test. Among the three different performance metrics that they used, we compare our results with recall since it is the closest one to our performance measure. They also reported overall accuracy results but they did not clarify the percentage of test used to obtain them.

Therefore, in the following sections we compare our results with the results of Shihavuddin *et al.* [SGG+13], Ani Brown Mary and Dharma [AD17], Ani Brown Mary and Dejey [AD18b] and Ani Brown Mary and Dejey [AD18a].

9.6.1.1 *Results Using Transfer Learning from ImageNet*

The results of Shihavuddin's method, Inception v3, ResNet with 50 and 152 layers and DenseNet with 121 and 161 layers are shown in Table 9.2, while the corresponding best configurations and execution times are shown in Table 10.4. Shihavuddin's model was executed on the CPU while the five CNNs were executed on the GPU. As it can be seen from these tables, ResNet-152 outperforms Shihavuddin's model and the rest of the CNN models. Inception provides a better accuracy than Shihavuddin's method on RSMAS, but shows a worse accuracy on EILAT. DenseNet shows the worst results on both datasets. In general, these results show that CNNs are able to become the state-of-the-art in coral classification tasks. In RSMAS, the best model is ResNet-152, with more than a 5% improvement with respect to Shihavuddin's method. In EILAT, ResNet-50 and ResNet-152 achieve exactly the same accuracy (this is not a cause of rounding), and they outperform Shihavuddin's method for more than 2%.

As we can see from Table 10.4, in general, the training process of the CNNs takes similar execution times. In particular, smaller batch sizes and larger numbers of epochs imply larger execution times. Besides, the training process of complex CNNs, such as DenseNet-161, takes slightly more time than the rest of CNNs when using the same hyperparameters.

In EILAT, Ani Brown Mary and Dharma [AD17] reported a recall of 96.4% using a test of 10% of the images and 84.54% using a test of 25% of the images. Ani Brown Mary and Dejey [AD18b] obtained a recall of 96.8% using a test of 10% and 87.1% using 25% of the images. Lastly, Ani Brown Mary and Dejey [AD18a] obtained 99.57% recall using a test of 10% and 87.49% using a test of 25%. In all cases, the results obtained with a test of 25% are lower than our results for EILAT, obtained training with 80% of the images and testing with the other 20% in each fold. On the other hand, the results obtained using a test of 10% are higher than our results, as you would expect since they are training with 90% of the images. However, it is interesting that our results are much closer to the ones obtained using a 10% test than to the ones obtained using a 25% test. Among the three works of Mary & Dejey, [AD18a] obtained the best results for EILAT. With this work, we have a difference of 10.36% with the 25% test and only 1.75% with the 10% test. The comparison can be seen in Table 9.4.

In RSMAS, Ani Brown Mary and Dharma [AD17] achieved a recall of 98.87% using a test of 10% of the images and 84.9% using a test of 25% of the images. Ani Brown Mary and Dejey [AD18b] reported a recall of 98.1% using a 10% test and

Table 9.4: Comparison between the results obtained by Ani Brown Mary and Dejey [AD18a] and our results for EILAT and RSMAS datasets.

Dataset	Mary & Dejey 10% test	Mary & Dejey 25% test	Our results (20% test)
EILAT	99.57	87.49	97.85
RSMAS	99.34	85.8	97.95

85.72% using a 25% test. Lastly, Ani Brown Mary and Dejey [AD18a] achieved 99.34% recall using a 10% test and 85.8% using a 25% test. As it happened with EILAT, Ani Brown Mary and Dejey [AD18a] obtained the best results among these three works and again our results using a test of 20% of the images are much closer to the ones obtained using a test of 10% than to the ones obtained using a test of 25%. With this work, we have a difference of 12.15% with respect to the 25% test and only 1.39% with respect to the 10% test. The comparison for RSMAS can also be seen in Table 9.4.

Obtained results allow us to conclude that only by training the last layers of a CNN that is already pre-trained on ImageNet and without data augmentation, which is the technique that usually takes more time, we can outperform a method that takes long running times and need high human supervision as it is the case with Shihavuddin's method. In fact, Shihavuddin's method is composed of six steps and each step is composed of one or various algorithms. Then, in order to obtain the best performance, it is needed to evaluate all the possible algorithm combinations through all the steps and to optimize the hyperparameters of each algorithm. Furthermore, this has to be done independently for each dataset we want to classify. In the case of Mary & Dejey's methods, the obtained differences between the results, combined with the fact that cross validation results are more stable than held-out results as we are using the mean over five experiments instead only one, allow us to conclude that our approach generates more precise models.

9.6.1.2 *Results Addressing the Class Imbalance Problem*

As we saw in Table 10.3, both EILAT and RSMAS are imbalanced. We address this problem by using a cost sensitive loss function as follows: We multiply the error produced by the model with a factor that depends on the proportion of the input image class with respect to the rest of classes. For example, if the network missclassifies an image that belongs to a class which is twice smaller than the rest of the classes, the error will be multiplied by a factor 2. In general, we assign a factor 1

Table 9.5: The accuracies obtained by Inception v3, ResNet-50, ResNet-152, DenseNet-121 and DenseNet-161 without data augmentation, using the set of hyperparameters in Table 10.4 and with a cost-sensitive loss function.

	Inception v3	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
EILAT	95.78	96.67	**97.50**	77.04	90.67
RSMAS	96.30	**97.67**	97.40	90.00	88.77

Table 9.6: Execution times, in minutes, of the experiments from Table 9.5

	Inception v3	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
EILAT	7.30	3.14	3.62	4.14	8.85
RSMAS	9.02	4.68	3.97	5.64	5.32

to the majority class and a factor $\frac{n_m}{n_c}$ to class c, where n_m is the number of images in the majority class and n_c is the number of images in class c, for $c = 1, \ldots, k$, where k is the total number of classes.

The impact of using cost-sensitive loss function along with transfer learning from ImageNet on the CNNs' performance is provided in Table 9.5, and the corresponding execution times are shown in Table 9.6. The obtained accuracies are in general slightly lower than the accuracies without addressing the imbalance issue (see Table 9.2). In particular, the best accuracy in each dataset is obtained without the cost sensitive loss function. This can be probably explained by the fact that CNNs are tolerant to such imbalance level: the majority of the factors are below 4, and there is only one class that has a factor of 12.17. These ratios are not considered high when using CNNs [BMM18]. If we compare the execution times in Tables 10.4 and 9.6, they are extremely similar, as we are training with the same number of images during the same number of epochs and using the same batch size in each case.

As a consequence, we choose not to address the imbalance problem in these datasets and therefore we choose to use the regular loss function.

9.6.1.3 *Results Using Transfer Learning from ImageNet and Fine-Tuning the Weights with MLC-2008*

Since EILAT and RSMAS are too small to fine-tune all the weights in the networks, we have used MLC-2008, which has 43,832 images, to fine-tune the network weights. That

Table 9.7: The accuracies obtained by Inception v3, ResNet-50, ResNet-152, DenseNet-121 and DenseNet-161 without data augmentation, using the set of hyperparameters in Table 10.4 and fine-tuning the networks with MLC-2008.

	Inception v3	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
EILAT	93.63	**95.87**	95.25	95.07	94.98
RSMAS	89.59	**95.34**	94.66	94.79	94.52

Table 9.8: Execution times, in minutes, of the experiments from Table 9.7. The first term in the sum is the time corresponding to fine-tune the network with MLC-2008 and the second term is the time corresponding to train the last two-fully connected layers with EILAT and RSMAS.

	Inception v3	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
EILAT	1110.34 + 8.51	915.74 + 4.25	2737.65 + 6.77	1505.07 + 6.81	3264.03 + 12.99
RSMAS	1110.34 + 9.91	915.74 + 6.05	2737.65 + 7.10	1505.07 + 8.88	3264.03 + 8.88

is, we have used the networks already pre-trained on ImageNet, then we have fine-tuned their weights with MLC-2008 and finally we have added two fully connected layers which are the ones that we have fine-tuned on RSMAS and EILAT.

In this experiment, we have trained all the networks using 100 epochs. We have used a batch size of 32 for Inception, ResNet-50 and DenseNet-121, and a batch size of 16 for ResNet-152 and DenseNet-161. To fit the largest CNN into the memory of the GPU, ResNet-152 and DenseNet-161, we have used a batch size of 16.

The obtained accuracies in this experiment are shown in Table 9.7 and the corresponding execution times are shown in Table 9.8. There are two important things to note from these results. First, the best result for each dataset is lower than the best result obtained without fine-tuning the networks with MLC-2008, see Table 9.2. Second, the only network that has improved with fine-tuning from MLC-2008 is DenseNet. This can be explained by the fact that the large volume of connections in DenseNet makes the network much more adapted to ImageNet than the rest of CNNs. This makes DenseNet less competitive when fine-tuned on EILAT and RSMAS than the other evaluated CNNs. Besides, as shown in Table 9.8, the time needed to fine-tune all the layers of DenseNet on MLC-2008 increased drastically in comparison with the rest of CNNs.

Table 9.9: The accuracies and execution times in minutes obtained by the best performing CNN on EILAT, ResNet-50, with different data augmentation techniques using the set of hyperparameters indicated in Table 10.4. The best result is stressed in bold.

	without data augmentation	shift = 0.2	zoom = 0.2	rotation = 2	flip	shift = 0.2, zoom = 0.2
Accuracy	97.85	**98.03**	97.85	97.40	97.53	97.85
Time (min)	3.34	91.70	90.25	90.10	72.23	88.76

Table 9.10: The accuracies and execution times in minutes obtained by the best performing CNN on RSMAS, ResNet-152, with different data augmentation techniques using the set of hyperparameters indicated in Table 10.4. The best result is stressed in bold.

	without data augmentation	shift = 0.2	zoom = 0.4	rotation = 2	flip	shift = 0.2, zoom = 0.4
Accuracy	97.95	98.36	**98.63**	97.40	97.578	98.08
Time (min)	3.71	57.02	56.20	57.50	49.21	56.13

To sum up, the best result on EILAT and RSMAS without considering data augmentation have been obtained by ResNet-50 and ResNet-152, respectively, and without using a cost-sensitive loss function or fine-tuning with MLC-2008.

9.6.2 *Classification of Coral Texture Images with Data Augmentation*

In this subsection we have analyzed the effect of the data augmentation techniques listed in Section 9.5 on the classification of texture images. To keep our analysis brief and concise, we have evaluated the data augmentation techniques only on the best performing models. In EILAT, ResNet-50 and ResNet-152 are the best models and provide the same accuracy, so we have chosen ResNet-50 as it is simpler and has less parameters. In RSMAS, the best model is ResNet-152. In both cases, we have used the regular loss function without fine-tuning with MLC-2008.

Recall that if we note `rotation = 2`, it means that we are applying a random rotation to the images by an angle in the interval $[0, 2]$. This notation is equivalent for all the other techniques.

Tables 9.9 and 9.10 show respectively the results of ResNet-50 and ResNet-152 on EILAT and RSMAS using data augmentation together with the parameters that provide the best performance. The number of steps is the number of times that we generate a batch of new images by data augmentation at each epoch. As the number of steps increases, the accuracy improves but also the time needed to complete each experiment increases. In this case, we have used 300 steps. We have evaluated different parameters for each data augmentation technique and several combinations between them. In Tables 9.9 and 9.10 we show the ones that obtained better accuracies. The difference in accuracy between using the best data augmentation technique and without using data augmentation is quite small, less than 1% in both datasets. However, the execution times are more than 15 times higher.

This slight improvement using data augmentation can be explained by the nature of used images. Since the original images are small and close-up, the applied modifications do not have much effect on the learning of the models as they need to be small: the shift implies to loose part of the images, and they are already very small; and the zoom implies to loose quality of the images, and they are already blurry because they are underwater images. On the other hand, the images are so close-up that the rotation and the flipping do not introduce significant variations among them. Besides, the performance of the base models is already good without any data augmentation. Therefore, we can conclude that the use of data augmentation techniques in texture coral images does not significantly improve the learning model.

9.6.3 *Analyzing the Misclassified Images*

In this subsection we have analyzed the misclassified images in each partition of the 5 fold cross validation in both datasets, EILAT and RSMAS.

In EILAT, ResNet-50 produced 22 misclassified images in all of the test folds. 14 of this misclassified images have been classified as Dead Coral. Dead Coral is the class with the highest number of images as all the dead corals (no matter what species) are in this class. This implies that this class shares some features with all the other classes, as we can see in Figure 9.7. Similarly, there are four Dead Coral images classified as other classes. In total, 18 of 22 images are misclassified due to this class. The remaining three images are from the class Branches Type II misclassified as Branches Type III or vice versa. As we can observe in Figure 9.8, some images in these two classes are very similar and therefore it is very difficult to distinguish between them.

(a) Examples of five images misclassified as Dead Coral

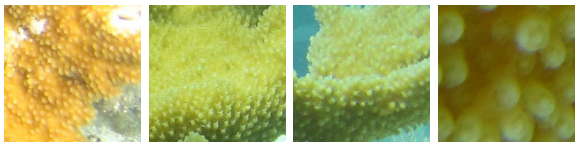
(b) Dead Coral images

Figure 9.7: Examples of (**a**) misclassified images in EILAT as Dead Coral and (**b**) original Dead Coral images.

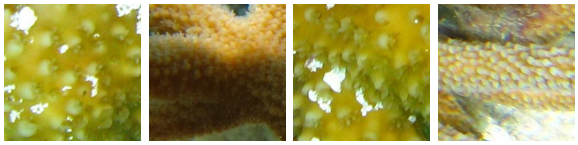
(a) Branches Type III images

(b) Branches Type II images

Figure 9.8: Examples that show the similarities between (**a**) Branches Type III and (**b**) Branches Type II. The third image form (**a**) is missclasified as Branches Type II. The first and second images from (**b**) are missclassified as Branches Type III.

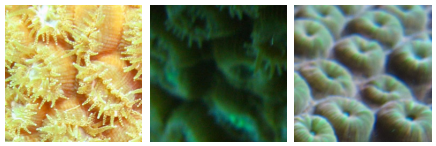


(a) APAL

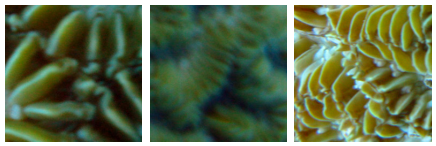


(b) ACER

Figure 9.9: Examples that show the similarities between (**a**) APAL and (**b**) ACER. The third and fourth images from (**a**) are missclassified as ACER. The first and second images from (**b**) are missclassified as APAL.



(a) MCAV



(b) MMEA

Figure 9.10: Examples that show the similarities between (**a**) MCAV and (**b**) MMEA. The second and third images in (**a**) are missclassified as MMEA.

In RSMAS, ResNet-152 produced only 10 misclassified images in all of the test folds. In general, the model tends to misclassify APAL as ACER and vice versa. The model always misclassified MCAV as MMEA. We can see the similarities between these classes in Figures 9.9 and 9.10. The rest of the misclassified images are blurry images.

From these misclassified images, we can conclude that in the case of EILAT it would be needed an expert to distinguish between the images in the class Dead Coral and the rest of the classes, as the images are very similar. For the images in Branches Type II and Type III a good solution might be for an expert to reclassify the images into more specific classes, like the coral species, which we have seen with RSMAS that is a good option. In the case of RSMAS, the misclassified images are again between classes that looks very similar between them, so we would need an expert to distinguish between them. In this case, as the images are so close-up, maybe it would be a good solution to make use of images from the same species that contain the whole coral body.

9.6.4 *Generalizing Our Approach to Other Coral Texture Datasets*

In order to see the generalization of our approach to other coral texture datasets, we have classified EILAT2, a subset of EILAT with 303 images of size 128×128 organized in five classes, and a randomly obtained subset of MLC-2008 with 36,500 images. We have chosen 36,500 images in order to make a fair comparison with the results reported in [AD17; AD18b; AD18a], since we cannot obtain the same dataset it was used in these works. In both cases, we have compared with all the works that used these two datasets. To do this, we have used the best CNN model for EILAT and RSMAS, respectively. We have used ResNet50 with the best hyperparameters for EILAT to classify EILAT2 and ResNet152 with the best hyperparameters for RSMAS to classify MLC-2008. In both cases we have used transfer learning from ImageNet and we have only trained the newly added two fully connected layers. For EILAT2 we have chosen the same network as EILAT since they are very similar datasets. For MLC-2008 we have chosen the same network as RSMAS guided by the number of classes in both datasets.

For EILAT2, we obtain an accuracy of 98.97% using a 5 fold cross validation. Shihavuddin *et al.* [SGG+13] reported an accuracy of 93.1%; Shakoor and Boostani [SB18] obtained an accuracy of 90.35%; Ani Brown Mary and Dharma [AD17] obtained

a recall of 99.1 using a test of 10% of the images and 87.43% using a test of 25% of the images; Ani Brown Mary and Dejey [AD18b] obtained a 97.12% recall using a 10% test and 87.4% using a 25% test; and Ani Brown Mary and Dejey [AD18a] achieved a recall of 99.12% with a 10% and a 88.69% using a 25% test. As we can see, our results outperformed the ones obtained by Shihavuddin *et al.* [SGG+13] and by Shakoor and Boostani [SB18]. For the other three works, our results outperformed the ones obtained with a test of 25% of the images but not the ones obtained using a test of 10%, as it occurred with EILAT and RSMAS. However, similarly to EILAT and RSMAS, our results are much closer to the ones obtained with a 10% test than the ones obtained with a 25% test. If we compare with the best results, the ones obtained by Ani Brown Mary and Dejey [AD18a], we have a difference of 10.28% with the results using a 25% test and a difference of 0.15% with the results using a 10% test. The small difference with the results using a 10% and the fact that our results were obtained with a 5 fold cross validation, which obtains more reliable and stable results, allow us to conclude that our approach is better for EILAT2. The comparison with this work is shown in Table 9.11.

For MLC, we have compared with all the works that used a version of this dataset. With a subset of 36,500 images of MLC-2008 we have obtained an accuracy of 76.66%. Beijbom *et al.* [BEK+12] obtained an accuracy of 74.3% using two thirds of MLC-2008 to train and the other third to test, an accuracy of 67.3% using MLC-2008 to train and MLC-2009 to test and an accuracy of 83.1% using MLC-2008 and MLC-2009 to train and MLC-2010 to test. Mahmood *et al.* [MBA+16b] used the same experimental framework and obtained accuracies of 77.9%, 70.1% and 84.5% in the three experiments, respectively. Shihavuddin *et al.* [SGG+13] used a subset of 18,879 images of MLC-2008 and obtained an accuracy of 85.5%. Shakoor and Boostani [SB18] used the same subset as in [SGG+13] and obtained an accuracy of 63.21%. Ani Brown Mary and Dharma [AD17] obtained a recall of 90.4% using a 10% test and 80.45% using a 25% test, Ani Brown Mary and Dejey [AD18b] reported recalls of 91.27% and 84.94%, respectively, and Ani Brown Mary and Dejey [AD18a] achieved recalls of 93.61% and 83.22%, respectively, using a subset of 36,500 images of the completed MLC. With this method, we outperformed some experiments in [BEK+12] and [MBA+16b], but in general our result is worse than the rest of the methods. The comparison with the best method [AD18a] is also in Table 9.11.

This is because the classification of EILAT2 and MLC20008 are very different cases from our approach point of view. We have shown that training only two added fully connected layers to a ResNet CNN model perform very well for small coral texture

Table 9.11: Comparison between the results obtained by Ani Brown Mary and Dejey [AD18a] and our approach for EILAT2 and MLC datasets.

Dataset	Mary & Dejey 10% test	Mary & Dejey 25% test	Our results (20% test)
EILAT2	99.12	88.69	98.97
MLC	93.61	83.22	76.66

datasets, as it is the case of EILAT, RSMAS and EILAT2. The results obtained for EILAT2 are similar to the ones obtained for EILAT and RSMAS, showing that our approach outperforms Mary & Dejey's approach on EILAT2. However, if the dataset is large enough to fine-tune the whole network, as it is the case with MLC-2008, the performance should improve doing so compared to training only two layers. To prove this we have fine-tuned the whole ResNet152 network with our subset of MLC-2008 and we have obtained an accuracy of 83.45%. As we can see, this result is closer to the rest of the works that used MLC and it is better than our result training only the last two layers of the network.

9.7 CONCLUSIONS

The classification of underwater coral images is challenging due to the large number of different coral species, the great variance among images of the same coral species, the lightning variations due to the water column, or the fact that several species tend to appear together, leading to an increasing overlapping among different classes. Few works have tackle this problem, but the only one that classifies EILAT and RSMAS is a really complex method which makes use of several algorithms and takes a lot of human intervention and time. We have addressed these problems by using some of the most powerful CNNs, namely Inception v3, ResNet and DenseNet. We have carried out a study of the foundations of this three CNNs, their parameter set-up, and the possibility of using fine-tuning from a related domain dataset and data augmentation techniques to aid their learning process. We have also studied the possibility of tackling the imbalance problem in the loss function. We have been able to outperform the state-of-the-art approach, proving that CNNs are an excellent technique for automatic classification of underwater coral images.

We have shown that CNNs based models achieved the state-of-the-art accuracies on the coral datasets RSMAS and EILAT, surpassing classical methods that require a

high human intervention, and without using data augmentation. In particular, ResNet have been the best CNN in RSMAS and EILAT.

We have shown that when the imbalance ratios are not too high, there is no improvement in the use of a cost-sensitive loss function. In addition, we have shown that if the datasets are small, a simpler network, like ResNet-50, performs better than a more complex network, like DenseNet-121 or DenseNet-161 even when using a previous fine-tuning with a larger related domain dataset.

When considering the impact of data augmentation, we have shown that from these two datasets, which contain very close-up images taken under similar conditions and have a lot of inner-class variance, there is a little benefit obtained from using such techniques.

This work enables new advanced challenges like classifying not just texture coral images, but structure coral images too. In particular, the problem of classifying any coral image using a single classifier, either texture or structure, will be addressed.

ACKNOWLEDGMENTS

This work was partially supported by the Spanish Ministry of Science and Technology under the project TIN2017-89517-P and by the Andalusian Government under the project P11-TIC-7765. Siham Tabik was supported by the Ramón y Cajal Programme (RYC-2015-18136) and Anabel Gómez-Ríos was supported by the FPU Programme 998758-2016 and by a scholarship of initiation to research granted by the University of Granada. The NVIDIA Titan Xp used for this research was donated by the NVIDIA Corporation.

REFERENCES

[ARV+17] C. Affonso, A. L. D. Rossi, F. H. A. Vieira and A. C. P. de Leon Ferreira de Carvalho, 'Deep learning for biological image classification', *Expert Systems with Applications*, vol. 85, pp. 114–122, 2017.

[AD18a] N. Ani Brown Mary and D. Dejey, 'Coral reef image/video classification employing novel octa-angled pattern for triangular sub region and pulse coupled convolutional neural network (pccnn)', *Multimedia Tools and Applications*, Jun. 2018.

[AD18b] N. Ani Brown Mary and D. Dejey, 'Classification of coral reef submarine images and videos using a novel z with tilted z local binary pattern (z+tzlbp)', *Wireless Personal Communications*, vol. 98, no. 3, pp. 2427–2459, Feb. 2018.

[AD17] N. Ani Brown Mary and D. Dharma, 'Coral reef image classification employing improved ldp for feature extraction', *Journal of Visual Communication and Image Representation*, vol. 49, pp. 225–242, 2017.

[BEK+12] O. Beijbom, P. J. Edmunds, D. I. Kline, B. G. Mitchell and D. Kriegman, 'Automated annotation of coral reef survey images', in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 1170–1177.

[BTK+16] O. Beijbom, T. Treibitz, D. I. Kline, G. Eyal, A. Khen, B. Neal, Y. Loya, B. G. Mitchell and D. Kriegman, 'Improving automated annotation of benthic survey images using wide-band fluorescence', *Scientific reports*, vol. 6, p. 23 166, 2016.

[BFF+15] M. Bewley, A. Friedman, R. Ferrari, N. Hill, R. Hovey, N. Barrett, E. M. Marzinelli, O. Pizarro, W. Figueira, L. Meyer *et al.*, 'Australian sea-floor survey data, with images and expert annotations', *Scientific data*, vol. 2, p. 150 057, 2015.

[BMM18] M. Buda, A. Maki and M. A. Mazurowski, 'A systematic study of the class imbalance problem in convolutional neural networks', *Neural Networks*, vol. 106, pp. 249–259, 2018.

[Cho+15] F. Chollet *et al.*, *Keras*, https://github.com/keras-team/keras, 2015.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'Imagenet: A large-scale hierarchical image database', in *Computer Vision and Pattern Recognition (CVPR), 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[Ela15] M. Elawady, 'Sparse coral classification using deep convolutional neural networks', *arXiv preprint arXiv:1511.09067*, 2015.

[ESI17] *Endangered species international*, http://www.endangeredspeciesinternational.org/, Accessed on 13-02-2018, 2017.

[FBS+14] F. Ferrario, M. W. Beck, C. D. Storlazzi, F. Micheli, C. C. Shepard and L. Airoldi, 'The effectiveness of coral reefs for coastal hazard risk reduction and adaptation', *Nature communications*, vol. 5, p. 3794, 2014.

[FG17] A. Ferreira and G. Giraldi, 'Convolutional neural network approaches to granite tiles classification', *Expert Systems with Applications*, vol. 84, pp. 1–11, 2017.

[FCN+18] M. D. Ferreira, D. C. Corrêa, L. G. Nonato and R. F. de Mello, 'Designing architectures of convolutional neural networks to solve practical problems', *Expert Systems with Applications*, vol. 94, pp. 205–217, 2018.

[HZR+16] 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.

[HLV+17] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, 'Densely connected convolutional networks', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[IUCN17] *Iucn red list table of number of threatened species by major groups of organisms*, http://cmsdocs.s3.amazonaws.com/summarystats/2017-3_Summary_Stats_Page_Documents/2017_3_RL_Stats_Table_1.pdf, Accessed on 13-02-2018, 2017.

[KSH12] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[LBB+98] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[MBA+16a] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. Fisher, 'Automatic annotation of coral reefs using deep learning', in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–5.

[MBA+16b] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. Fisher, 'Coral classification with hybrid feature representations', in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 519–523.

[MAP+15] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[PRJ+08] O. Pizarro, P. Rigby, M. Johnson-Roberson, S. B. Williams and J. Colquhoun, 'Towards image-based marine habitat classification', in *OCEANS 2008*, IEEE, 2008, pp. 1–7.

[PAH+15] M. S. Pratchett, K. D. Anderson, M. O. Hoogenboom, E. Widman, A. H. Baird, J. M. Pandolfi, P. J. Edmunds and J. M. Lough, 'Spatial, temporal and taxonomic variation in coral growth—implications for the structure and function of coral reef ecosystems', *Oceanography and Marine Biology: An Annual Review*, vol. 53, pp. 215–295, 2015.

[RDS+15] O. Russakovsky *et al.*, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[SB18] M. H. Shakoor and R. Boostani, 'A novel advanced local binary pattern for image-based coral reef classification', *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2561–2591, 2018.

[SGG+13] A. Shihavuddin, N. Gracias, R. Garcia, A. C. Gleason and B. Gintert, 'Image-based coral reef classification and thematic mapping', *Remote Sensing*, vol. 5, no. 4, pp. 1809–1841, 2013.

[Shi17] A. Shihavuddin, *Coral reef dataset, v2.* Mendeley data https://data.mendeley.com/datasets/86y667257h/2, Accessed on 12-02-2018, 2017.

[SZ14] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556*, 2014.

[SD09] M. D. Stokes and G. B. Deane, 'Automated processing of coral reef benthic images', *Limnol. Oceanogr.: Methods*, vol. 7, no. 157, pp. 157–168, 2009.

[SLJ+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, 'Going deeper with convolutions', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[SVI+16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, 'Rethinking the inception architecture for computer vision', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[Yu17] F. Yu, *Resnet and densenet cnns in keras*, https://github.com/flyyufelix/cnn_finetune, 2017.

10

CORAL SPECIES IDENTIFICATION WITH TEXTURE OR STRUCTURE IMAGES USING A TWO-LEVEL CLASSIFIER BASED ON CONVOLUTIONAL NEURAL NETWORKS

Gómez-Ríos, A., Tabik, S., Luengo, J., Shihavuddin, A. S. M., & Herrera, F. (2019). Coral species identification with texture or structure images using a two-level classifier based on Convolutional Neural Networks. *Knowledge-Based Systems*, 184, 104891. DOI: https://doi.org/10.1016/j.knosys.2019.104891

- Status: Published
- Impact Factor (JCR 2019): 5.921
- Subject Category: Computer Science, Artificial Intelligence. Ranking 15/137 (Q1)

CORAL SPECIES IDENTIFICATION WITH TEXTURE OR STRUCTURE IMAGES USING A TWO-LEVEL CLASSIFIER BASED ON CONVOLUTIONAL NEURAL NETWORKS

Anabel Gómez-Ríos[a], Siham Tabik[a], Julián Luengo[a], ASM Shihavuddin[b], Francisco Herrera[a]

[a] Andalusian Research Institute in Data Science and Computational Intelligence, Dept. of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
[b] Dept. of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Kgs. Lyngby, Denmark

ABSTRACT

Corals are crucial animals as they support a large part of marine life. The automatic classification of corals species based on underwater images is important as it can help experts to track and detect threatened and vulnerable coral species. However, this classification is complicated due to the nature of coral underwater images and the fact that current underwater coral datasets are unrealistic as they contain only texture images, while the images taken by autonomous underwater vehicles show the complete coral structure. The objective of this paper is two-fold. The first is to build a dataset that is representative of the problem of classifying underwater coral images, the StructureRSMAS dataset. The second is to build a classifier capable of resolving the real problem of classifying corals, based either on texture or structure images. We have achieved this by using a two-level classifier composed of three ResNet models. The first level recognizes whether the input image is a texture or a structure image. Then, the second level identifies the coral species. To do this, we have used a known texture dataset, RSMAS, and StructureRSMAS.

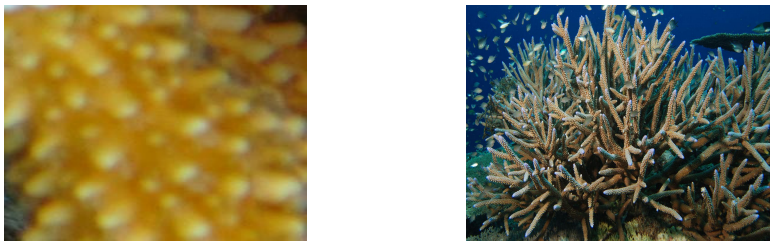
Keywords: Coral Images Classification, Structure Coral Images, Deep Learning, Convolutional Neural Networks, Inception, ResNet, DenseNet.

10.1 INTRODUCTION

Coral reefs are extremely valuable ecosystems for marine life and humans. A recent evaluation of 40% of the total number of coral species has shown that more than 200 species are threatened [IUCNb]. This is a direct consequence of air and water pollution and the changes in ocean temperatures due to climate change [SP16].

The automatic classification of corals based on images is a hard task. There are thousands of coral species and their taxonomy is continuously updated as more information is obtained. In addition, some coral species look really similar externally whereas their differences are based on internal characteristics. This makes difficult to keep a constant record of all the species, even more their extension rates. Nowadays, thousands of images of coral reefs and other benthic habitats are being captured regularly by Autonomous Underwater Vehicles (AUVs). However, analysing that huge amount of data and acquiring useful information out of it is still a bottleneck, as many hours of experts manual work are involved in such tasks. Having a rigorous accurate automatic coral classifier can potentially help in analysing the large amount of data, thereby progressing in the understanding of coral reefs. Nevertheless, the available coral datasets contain, in general, texture images, as opposed to the structure images obtained by AUVs. This is a consequence of using classical machine learning models, since they need to extract textures as a previous feature extraction step. Texture images show only a small part of the coral and do not include any information of the whole structure of the coral body, while structure images contain the whole coral or a large part of it. Figure 10.1 shows an example of (a) a coral texture or local information and (b) a coral structure or global information. The available public coral datasets, such as EILAT (which contains images taken near Eilat) RSMAS (Rosenstiel School of Marine and Atmospheric Sciences)or MLC (Moorea Labelled Corals), among others, only contain texture images. As far as we know, there is no publicly available datasets containing coral structure images.

Making the coral classification automatic has been tackled in previous works, using either normal machine learning models combined with feature extraction methods [BEK+12; PRJ+08; SGG+13; SD09] or Convolutional Neural Networks (CNNs) [Ela15; MBA+16a; MBA+16b; GTL+19]. Recently, CNNs are providing outstanding performance in pattern recognition in many fields, particularly in Computer Vision [KSH12; BBA16; LYJ+17]. A clear example is the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) competition [RDS+15], whose top ten models have been CNNs since 2012. A key to its success is that they are capable to extract simple and complex



(a) Texture (b) Structre

Figure 10.1: Difference between (a) coral texture and (b) coral structure.

features as the network goes deeper [SLJ+15; SVI+16]. In fact, since the last few years, they are capable to use the simple features in the deeper layers by adding some connections that skip layers [HZR+16; HLV+17]. Furthermore, they are able to overcome the limitation of large datasets requirements in the training phase by increasing the size of the training set artificially (data augmentation) or by starting the training using the weights from the pre-trained network on another dataset (transfer learning).

However, none of the previous works address the problem of classifying coral texture and structure images together, so it is still an unsolved issue. Most CNN works analyse classical architectures such as LeNet [LBB+98] and VGGnet [SZ14]. In [GTL+19], the authors analysed more recent CNNs, like Inception [SVI+16], ResNet [HZR+16] and DenseNet [HLV+17] on two texture datasets, RSMAS and EILAT. They achieved the state-of-the-art accuracies in both datasets. In general, CNNs outperform classical feature extraction in the classification of coral texture images.

In this work, we aim at identifying coral species based on their texture or structure images. To the best of our knowledge, this is the first work classifying corals based on texture and structure images. We propose to use recent CNNs to classify any coral image, either texture or structure. We want to provide a classifier that could be used with pictures provided by AUVs, irrespective of the portion of coral that is contained in each image. In order to achieve this, we have used a known texture dataset, RSMAS [Shi17], and a new structure dataset, more realistic, that we have built in this work, called StructureRSMAS. This dataset is available through the following link: `http://sci2s.ugr.es/CNN-coral-image-classification`. We have taken advantage of the costly experimentation that it has been done in [GTL+19] to choose the best model and configuration for RSMAS. For StructureRSMAS, we have

evaluated Inception, ResNet and DenseNet, and we have chosen the best model and its best configuration. We have also tested the influence of some image enhancement techniques on both datasets, RSMAS and StructureRSMAS, in order to improve the images before using the CNN models.Finally, we have built a two-level classifier, whose first level is to decide whether the input coral image is a texture or a structure and whose second level is to identify the coral species. Similar strategies have also been used in other works [XWB+18].To obtain the model in the first level, we have also evaluated the three CNN architectures mentioned before. To the best of our knowledge, this work is the first identifying coral species based on its local or global information.

The rest of the paper is organized as follows. In Section 10.2, we give an overview on CNNs and the three architectures we have used. In Section 10.3, a summary on previous works for classifying underwater coral images is given. In Section 10.4, we present the structure dataset we have created. In Section 12.3, we describe our proposal to classify coral images, either texture or structure and in Section 12.5 we explain the experiments we have carried out to obtain such a classifier. Finally, in Section 10.7 we state the conclusions of this work.

10.2 CONVOLUTIONAL NEURAL NETWORKS (CNNS) AND IMPROVEMENT TECHNIQUES

CNNs are a widely used type of artificial neural network. They constitute the state-of-the-art in object recognition in images. Using CNNs we do not have to extract the features from the images with a previous algorithm. This is possible thanks to its principal operation: the *convolution*, which is defined, for a point (i, j), as [GBC16]:

$$\sum_m \sum_n I(i-m, j-n)K(m,n) \ , \tag{10.1}$$

where I is the input of the convolution and K is the kernel of the convolution with size $m \times n$. Each convolutional layer has different numbers of kernels. As we can see in Figure 10.2, the input of the convolutional layer is convoluted with each kernel, resulting in an output feature map per kernel. The values of these kernels, called weights, are learned by the CNN autonomously during the training process. At the end of the training, these values are the selected features of the images.

To increase the non-linearity of the models, every convolutional layer is followed by a non-linear operation, typically the Rectified Linear Unit (ReLU) operation.

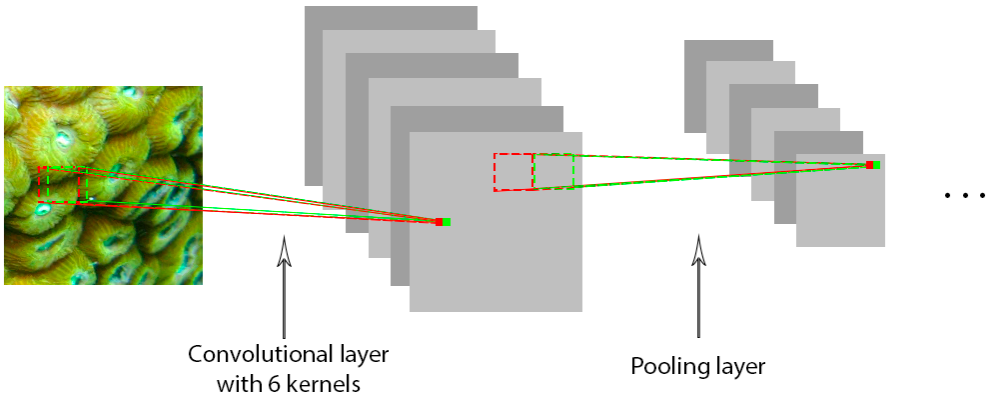


Figure 10.2: An example of a convolutional layer and a pooling layer in a CNN.

To increase the abstraction level of the extracted features, it is usual to reduce the size of the feature maps from the convolutional layer using a *pooling* layer, which takes a neighbourhood of size $m \times n$ and performs an operation to it. There are several variants, depending on the operation used: maximum pooling, minimum pooling, average pooling, etc. The size is reduced, as we can see in Figure 10.2, by using a stride between one group of pixels and the next one. Among other things, pooling allows the network to extract simple features at the beginning and complex features as the network goes deeper.This is why the first improvement to obtain better results with CNNs was to make the networks deeper. However, it has the problem that the deeper the network, the harder it is to train it. On the other hand, it is usual that, at the same level, the network has various convolutional layers, increasing the width of the network and extracting different features. This is clearly seen in Inception v3 architecture [SVI+16].

In the last years, some architectures, like ResNet [HZR+16] and DenseNet [HLV+17], have explored other ways to improve the performance of the networks. In particular, they added connections that skip layers, making the network to use the simple features extracted at the beginning in deeper layers. This has been proved to work well: ResNet won the ILSVRC competition [RDS+15] in 2015 and DenseNet beat its results in 2016.

In order to obtain good results from a CNN, it is necessary to train it with large datasets. However, there are two improvement techniques that help to overcome this constraint: transfer learning and data augmentation.

By using *transfer learning* we can start the training of the network from the pre-trained weights in another problem, although it is recommended that this other problem is somehow related with the problem that we want to resolve. Then, we can choose between retraining only the last layer of the network, which classifies the images into our classes, or to retrain all the weights in the network (or some of them), which is also called fine-tuning.

Data augmentation allow us to artificially increase the size of the training set, so we have more images to train the network. The increase in the training set is done by applying several distortions to the original images, like zooming them, flipping them horizontally or vertically, rotating them, shifting them, etc. They can be applied alone or combined, and in most of them we can choose how much distortion we want to apply: rotate an image 15 degrees, zoom 25% of the image, etc. Depending on the original images, we need to be careful with the data augmentation techniques we apply as the images may lose their meaning: for example, flipping the images in hand-writing digit classification.

In this work, we will use Inception v3, ResNet and DenseNet. These three architectures are based on the repetition of a block, different in each case. The composition of these networks is shown in Table 10.1. As it is seen in the table, Inception v3 has a fixed number of layers. That is because it has a fixed number of blocks, although the base inception module is only used at the beginning of the network and it has several modifications as the network goes deeper. On the other hand, the number of layers in ResNet and DenseNet is a hyperparameter and it depends on the number of times their blocks are repeated. There are two more hyperparameters in the three architectures: the number of epochs and the batch size we used to train them.

10.3 RELATED WORK ON CORAL CLASSIFICATION

In this section we first present the available coral datasets that have been used in the literature. Second, we present the previous works on the classification of coral images and the results obtained for RSMAS, the texture dataset we have used.

10.3.1 *Coral datasets*

The current available eight open datasets are KTH-TIPS (Textures under varying Illumination, Pose and Scale), CURET (Columbia-Utrecht Reflectance and Texture),

Table 10.1: Description of the composition of Inception v3, ResNet and DenseNet. BN stands for Batch Normalization.

Architecture	**Name of the block**	**Composition of the block**	**Same block along the network?**	**Number of layers of the network**
Inception v3	Base inception module	Three 1×1 convolutions, four 3×3 convolutions and one pooling. Some of these operations are made in parallel, so it has a concatenation filter at the end. All the convolutions are followed by BN and a ReLU.	No	42
ResNet	Building block	Three consecutive operations: 1×1 conv., 3×3 conv., and 1×1 conv. It has an additional connection between the input of the first 1×1 conv. and the output of the second 1×1 conv that performs a 1×1 conv. or an identity. All the convolutions are followed by BN and a ReLU.	Yes	It depends on the number of times the block is repeated.
DenseNet	Dense block	A repetition of the sequence: BN, ReLU, 1×1 conv., BN, ReLU and 3×3 conv. The output of all the layers inside the block is connected with the input of all the following layers in the block.	Yes	It depends on the number of times the block is repeated and the repetitions inside each block.

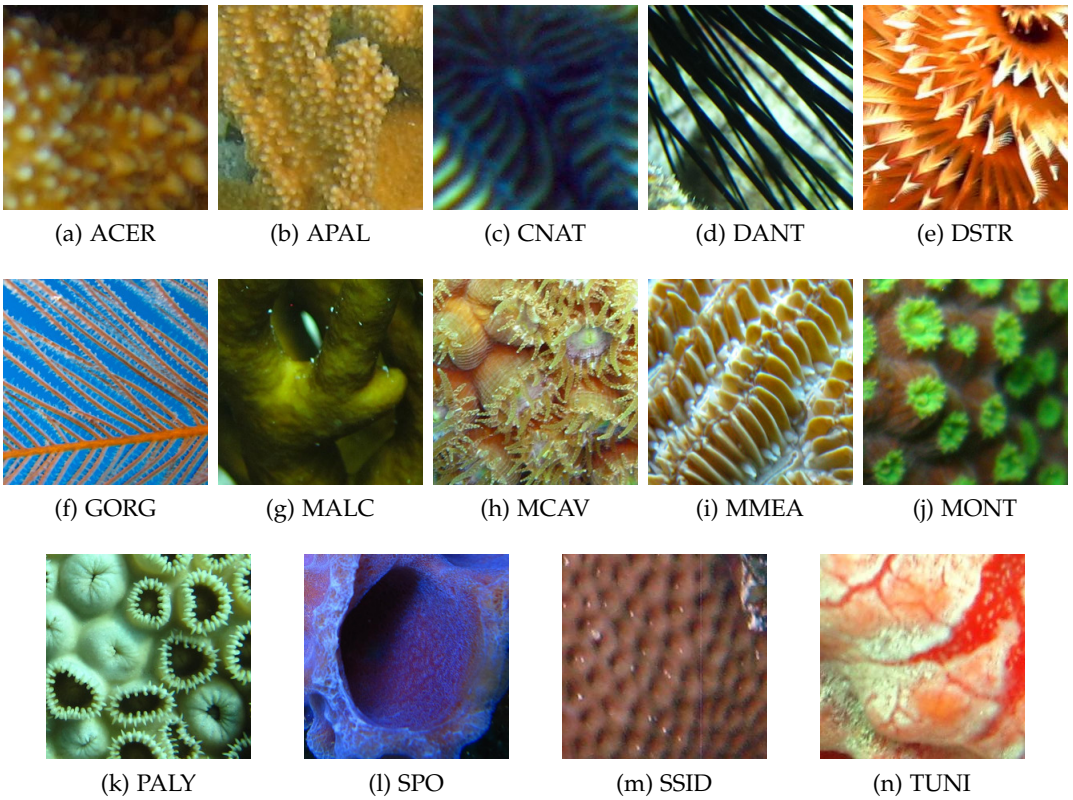
UIUCtex (University of Illinois at Urbana-Champaign texture dataset), MLC (Moorea Labelled Corals), EILAT, EILAT2, RSMAS (Rosenstiel School of Marine and Atmospheric Sciences)and the Red Sea Mosaic. Only MLC, EILAT, EILAT2, RSMAS and the Red Sea Mosaic contain RGB images. From these, RSMAS and EILAT are the ones with more number of classes, 14 and 8, respectively, and the most recent. EILAT2 is a subset of EILAT, MLC only contains five coral classes and the Red Sea Mosaic is actually a large image containing different coral species.

The images in all these datasets share some characteristics inherent to underwater images: the water movement cause lightning variations between images taken at the same time; the water causes the images to be blurry; and it is very common that animals, like fish, cover part of the corals when the images are being taken. On the other hand, the images share some characteristics inherent to coral images, like the occurrence of various coral species in the same image or the subjective classification of the images by different experts.

Both RSMAS and EILAT are texture datasets, but we choose RSMAS as our texture dataset because the labelling is the scientific Latin name of the coral species, which allow us to obtain structure images from these species and to create StructureRSMAS, a coral structure dataset that contains images from the same species that RSMAS. RMSAS contains 766 images that are patches that contain specific and small parts of the corals, not the entire structure of them. They are close-up images, so they are sometimes blurry and small: each one of them have 256×256 pixels in size. Some images from RSMAS can be seen in Figure 10.3.

10.3.2 *Previous works*

Before recent advances in deep learning, most previous approaches on the automatic coral reef classification with underwater images combined classical machine learning models with feature extraction algorithms [MSS05; SD09; BEK+12; SGG+13]. They used, in general, an algorithm to extract color features and another one to extract texture features. Then, they used them to train a classical machine learning model, like Support Vector Machines in [BEK+12] or a three layer neural network in [MSS05]. Most of them used a single dataset, usually small and containing several non-coral classes, except for [SGG+13], where the authors designed an algorithm that could be used with different datasets. They divided their algorithm into steps and at each



(a) ACER (b) APAL (c) CNAT (d) DANT (e) DSTR

(f) GORG (g) MALC (h) MCAV (i) MMEA (j) MONT

(k) PALY (l) SPO (m) SSID (n) TUNI

Figure 10.3: One texture image from each RSMAS class.

step several sub-algorithms could be chosen in accordance with the dataset that was going to be classified.

The authors in [AD17; AD18b; AD18a] proposed three different modifications of the local binary pattern feature descriptor, and they used the extracted features to train machine learning models in order to classify several datasets.

In 2015, the author in [Ela15] used CNNs for the first time to resolve this task. He used a LeNet-5 model, but he still combined it with feature extraction algorithms. Since then, most works classifying coral images have used CNNs, as they provide better results and do not need a previous step of feature extraction. The authors in [MBA+16a] and [MBA+16b] used a VGGnet model and in [MBA+16b] the authors also combined the features extracted by the CNN model with hand-crafted features, although the improvement from such custom features was small. In [MBA+16a] they used the Benthoz15 dataset and in [MBA+16b] they used the MLC dataset.

In [GTL+19], the authors proposed the use of more powerful CNNs to classify EILAT and RSMAS [Shi17], and they found that ResNet was the best CNN architecture in both datasets, improving the state-of-the-art results obtained by [SGG+13; AD17; AD18b] and [AD18a], which were all the works that used EILAT and RSMAS.

The works that used RSMAS and the results they obtained can be seen in Table 10.2. The highest result is a recall of 99.34%, obtained by [AD18a], but using a held out test of 10%. The authors in [GTL+19] obtained an accuracy of 98.63% using a five fold cross validation technique, which means that they were using tests sets of 20% of the dataset in each partition. Then, they took the mean of the test accuracy on the five partitions. As a consequence, this result is more stable. Because of this, the best model for RSMAS, and the one we choose to use in this work, is the one obtained in [GTL+19].

An extensive review on the classification of coral images using deep learning can be found in [MBA+17]. However, most of these works used texture datasets whose images show close-up and very specific patches of the corals, not the whole structure of them. The present work is different from all the previously cited works in that it develops an automatic model capable of identifying the species of a coral based on an input image of either texture or structure.

Table 10.2: Results from previous works on RSMAS. The results of Shihavuddin et al. using a 5 fold cross validation can be found in [GTL+19].

Authors	Ref.	Metric	Result	Test method
Brown Mary et al.	[AD17]	Recall	98.87%	10% held out test
Brown Mary et al.	[AD17]	Recall	84.9%	25% held out test
Brown Mary et al.	[AD18b]	Recall	98.1%	10% held out test
Brown Mary et al.	[AD18b]	Recall	85.72%	25% held out test
Brown Mary et al.	[AD18a]	Recall	99.34%	10% held out test
Brown Mary et al.	[AD18a]	Recall	85.8%	25% held out test
Shihavuddin et al.	[SGG+13]	Accuracy	92.74%	5 fold cross validation
Gómez-Ríos et al.	[GTL+19]	Accuracy	98.63%	5 fold cross validation

10.4 STRUCTURERSMAS: A NEW CORAL STRUCTURE DATASET

In this section we present the new coral dataset of coral structure images we have built: StructureRSMAS. We have considered the same coral species, nomenclature and number of classes as in RSMAS. We have downloaded the images from official scientific websites, e.g., the Encyclopedia of Life [EOL], the IUCN Red List of Threatened Species [IUCNa] or the coralpedia of the University of Warwick [CUW]. Few images were available per class.

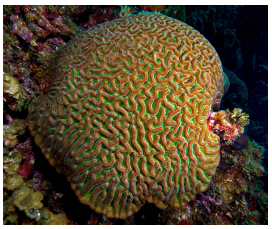
We have built StructureRSMAS because typical coral pictures often capture the whole coral body and structure and do not focus only on the texture of its parts. An ideal classifier should be able to recognize the coral based either on its texture or structure and with images taken under different conditions, cameras, etc. To develop such a model we need images that show both texture and structure of the coral species.

StructureRSMAS contains 409 coral images of variable size but larger than the ones in RSMAS. Some examples can be seen in Figure 10.4. The dataset, along with a list of the sources of the images, is available through the following link: `http://sci2s.ugr.es/CNN-coral-image-classification`. In Figure 10.3 and Figure 10.4 we can see the difference between texture and structure images from the same coral species.

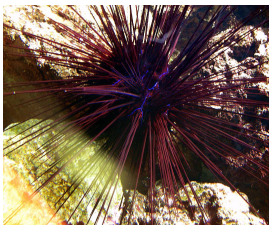
The images in RSMAS and StructureRSMAS were taken with different cameras and under different conditions. They are both imbalanced, as we show in Table 10.3,

(a) ACER

(b) APAL

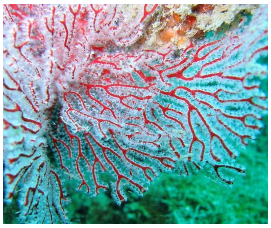


(c) CNAT



(d) DANT

(e) DSTR

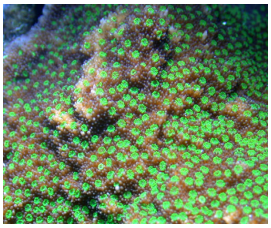


(f) GORG

(g) MALC

(h) MCAV

(i) MMEA



(j) MONT

(k) PALY

(l) SPO

(m) SSID

(n) TUNI

Figure 10.4: One structure image from each StructureRSMAS class.

Table 10.3: Characteristics of RSMAS and StructureRSMAS. #imgs refers to the number of images in that class.

Classes	**#imgs in RSMAS**	**#imgs in StructureRSMAS**
Acropora Cervicornis (ACER).	109	44
Acropora Palmata (APAL).	77	41
Colpophyllia Natans (CNAT).	57	34
Diadema Antillarum (DANT).	63	20
Diploria Strigosa (DSTR).	24	16
Gorgonians (GORG).	60	18
Millepora Alcicornis (MALC).	22	33
Montastraea Cavernosa (MCAV).	79	38
Meandrina Meandrites (MMEA).	54	30
Montipora spp. (MONT).	28	21
Palythoas Palythoa (PALY).	32	32
Sponge Fungus (SPO).	88	23
Siderastrea Siderea (SSID).	37	36
Tunicates (TUNI).	36	23

since there are classes that contain more images than others, like ACER in RSMAS, which has 109 images, while MALC has 22 images. The differences are smaller in StructureRSMAS. The largest difference in this dataset is between ACER, with 44 images, and DSTR, with 16 images.

10.5 A TWO-LEVEL CLASSIFIER FOR CORAL CLASSIFICATION USING A TEXTURE MODEL AND A STRUCTURE MODEL

We propose the use of a two-level classifier to address the automatic classification of corals based on either texture or structure images. This classifier is composed of three models, one used in the first level and the other two used in the second level. As we show in Figure 10.5, the first level is to determine if the image is a texture or a structure. Therefore, the model used in this level is a binary classier trained over the images of RSMAS (texture images) and the images of StructureRSMAS (structure images) as two separated classes. In the second level, we decide which coral species

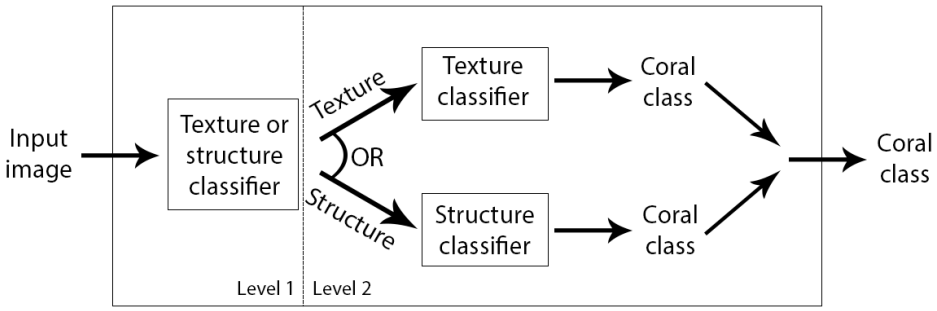


Figure 10.5: The two-level classifier we have developed to classify any coral image, either texture or structure.

the image belongs to using a texture model, trained over RSMAS, or a structure model, trained over StructureRSMAS, depending on the output obtained in the first level. To build our two-level model, at the first level we have developed a binary texture or structure classification model. At the second level, we have considered the most accurate texture model from the literature and developed a new structure classification model using StructureRSMAS.

In particular, for the texture dataset, RSMAS, we have used the state-of-the-art model proposed in [GTL+19], a ResNet-152 model that used transfer learning from ImageNet and data augmentation techniques.

For the structure dataset, StructureRSMAS, and the texture or structure binary classifier, we have evaluated different CNN architectures and configurations, as we will see in the following section.

In both datasets, we have evaluated the improvement of some image enhancement techniques before the application of the data augmentation techniques.

10.6 EXPERIMENTAL ANALYSIS

In this section we describe the process we have used to build the two-level classifier. First, we describe the experimental framework we have used in all the experiments.

Table 10.4: The set of hyperparameters we have test in the three architectures.

	Number of layers	Number of epochs	Batch size
Inception v3	42	100, 300, 500, 700, 1000, 1300	32, 64, 128
ResNet	50, 152		
DenseNet	121, 161		

Second, we evaluate different image enhancement techniques in the RSMAS dataset.Third, we analyse and compare different CNN architectures for StructureRSMAS and choose the best of them. We also evaluate the image enhancement and data augmentation techniques in this dataset, and choose the best configuration for it.Fourth, we evaluate and compare the same CNN architectures, and data augmentation techniquesto classify an image into texture or structure using the image enhancement techniques for both datasets that we found the best for them. Finally, we analyse and evaluate the performance of the proposed two-level classifier.

10.6.1 *Experimental framework*

The results shown in this section are obtained using a 5 fold cross validation technique. The known accuracy metric has been used to compare the performance of the CNNs and the data augmentation techniques.

For the implementation of Inception, ResNet and DenseNet, we have used Keras [Cho+15] with Tensorflow [MAP+15] as backend. For Inception, we have used the model already available in Keras 2.0.4, and we have adapted the code by Yu Felix in GitHub [Yu17] for ResNet and DenseNet.

To search for the best model for StructureRSMAS and for the decision between texture and structure, we have evaluated the hyperparameters that we show in Table 10.4 in a grid search.

Since RSMAS and StructureRSMAS are very small to train the CNNs from scratch, as we can see in Table 10.3, we have used transfer learning from ImageNet [DDS+09]. That way, we have all the networks with its weights pre-trained on ImageNet. Then, we have removed the last Fully Connected (FC) layer with 1000 neurons, which classifies the inputs into ImageNet classes, and we have added two FC layers, the first with 512 neurons and a ReLU activation and the second with as many neurons as classes in the dataset we are classifying (14 for StructureRSMAS and 2 for the

Table 10.5: Description of the evaluated data augmentation techniques.

Data augmentation technique	**Parameter**	**Description**
Shift	A float number x	To shift vertically and horizontally the images by a random fraction of the width or length, respectively, in $[0, x]$.
Zoom	A float number x	To zoom the images so their width and length are a random number in $[1-x, 1+x]$.
Rotation	An integer number x	To rotate the images by a random degree in $[0, x]$.
Flip	True/False	If true, randomly choose if the image is horizontally flipped or not.

classifier between RSMAS and StructureRSMAS) and a softmax activation. Lastly, we have only trained the two FC layers we have added.

Once we have chosen the best CNN model and its best parameters, we test the following image enhancement techniques: contrast and brightness enhancement (referred to later as CBE) [PAA+87; XTJ+18], saliency detection [ZMH16; SDB+14] and deblurring [TXL+18; TGS+18]. Then, we choose the best combination of them for each dataset and with it, we evaluate data augmentation techniques.We have evaluated the performance of data augmentation by carrying out experiments with and without the use of data augmentation techniques. The description of the data augmentation techniques we have used is in Table 10.5.

10.6.2 *Second level: texture model*

In this section we evaluate the image enhancement techniques in the RSMAS dataset. We use the best model and hyperparameters found in [GTL+19] for this dataset: a ResNet152 model with batch size 32 and 300 epochs for the training process. The results can be seen in Table 10.6, where CBE refers to contrast and brightness enhancement. As we can observe in this table, none of the image enhancement techniques helps to improve the accuracy of the model in RSMAS, though the

Table 10.6: Results obtained for RSMAS using ResNet-152 for each image enhancement technique. The best accuracy is stressed in bold.

	Without enhancement	CBE	Deblur	Saliency	Deblur + CBE	Deblur + CBE + Saliency
Accuracy	**96.710**	96.575	96.438	79.726	95.342	72.740

differences, if we exclude the saliency method, are similar. We argue that this is happening because the images in this dataset are already preprocessed. In the case of the saliency method, we think this is normal as we are loosing all the information in the background of the images.

As a result, we are not using any image enhancement techniques in this dataset and thus we are using the best data augmentation technique reported in [GTL+19], a random zoom of 0.4, which give us an accuracy of 98.356%.

10.6.3 *Second level: structure model*

To automatically classify any coral image from AUVs, it is necessary to classify images of entire corals. At this level, we focus on the problem of classifying structure images before solving the classification of texture and structure images together. These images are generally larger than the images in RSMAS. To simplify the number of experiments, we first have chosen the best model to classify the complete images in StructureRSMAS, among several CNNs models, without using data augmentation. After that, we have chosen the best model to evaluate the image enhancement methods and we have chosen the best of them. Once we have the best model and the best image enhancement method, we evaluate the data augmentation techniques.

The results without data augmentation and without image enhancement, and the hyperparameters we have used to obtain those results, are shown in Table 10.7. Similarly to RSMAS, ResNet is the best classifier. Particularly, ResNet-50 achieves the highest accuracy, although Inception also obtains a competitive accuracy. DenseNet is the only CNN that does not show a good performance in this problem. As stated in [GTL+19], this is probably due to the large number of connections in DenseNet, which makes it more adapted to ImageNet than the rest of the architectures. As a consequence, it is more difficult to classify a new dataset just by training the last two layers of the network.

Table 10.7: Best results obtained for StructureRSMAS using Inception, ResNet-50, ResNet-152, DenseNet-121 and DenseNet-161, and the set of hyperparameters used to obtain them, without data augmentation. The best accuracy is stressed in bold.

	Inception	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
Accuracy	81.316	**83.158**	83.158	54.737	55.526
Best batch size	32	32	32	64	32
Best number of epochs	700	300	1300	700	700

Table 10.8: Results obtained for StructureRSMAS using ResNet-50 for each image enhancement technique. The best accuracy is stressed in bold.

	Without enhancement	CBE	Deblur	Saliency	Deblur + CBE	Deblur + CBE + Saliency
Accuracy	82.368	83.684	**85.000**	52.632	83.158	51.358

We observe that the accuracy obtained for StructureRSMAS is lower than the one obtained for RSMAS. This can be explained by the fact that StructureRSMAS contains fewer examples than RSMAS, and the images that belong to the same class have very different characteristics e.g., resolution, angle of view or distance from which the images are captured. Despite all of this, we achieve a good accuracy, 83.158%.

Next, we have used the ResNet-50 model with its best parameters to test the image enhancement methods in this dataset. The results we have obtained are showed in Table 10.8, and we can see that for this dataset it is better to first preprocess the images by using contrast and brightness enhancement and deblurring. The highest accuracy with image enhancement, 2.6% higher than without applying any image enhancement, was obtained by applying the deblurring method alone. While applying the contrast and brightness enhancement improved the accuracy by only 1.3%. As a result, we have preprocessed the images in StructureRSMAS with the deblurring method.

Finally, we have evaluated different data augmentation techniques with the ResNet-50 classifier, and, for the sake of brevity, the best results can be seen in Table 10.9. They show that there is no improvement from using data augmentation in this dataset.We argue that this can be due to the very small size of the dataset, which also affects even when using data augmentation, as there are few images from which to obtain new images.

Table 10.9: Best results obtained for StructureRSMAS using ResNet-50 for each data augmentation technique. The best accuracy is stressed in bold.

	shift = 0.4	flip	zoom = 0.4	rotation = 6	flip + rotation = 8
Accuracy	84.211	**85.000**	83.421	84.737	83.684

Table 10.10: Best results obtained for the texture or structure binary classifier using Inception, ResNet-50, ResNet-152, DenseNet-121 and DenseNet-161 and the set of hyperparameters used to obtained them, without data augmentation. The best accuracy is stressed in bold.

	Inception	ResNet-50	ResNet-152	DenseNet-121	DenseNet-161
Accuracy	95.495	99.640	**99.730**	98.920	99.009
Best batch size	64	32	64	32	64
Best number of epochs	300	700	1000	1000	1300

10.6.4 *First level: texture or structure binary model*

In this section we develop a classifier to distinguish whether an image is a texture or a structure. To do this, we have joined in one dataset all the images in RSMAS in one class and all the images in StructureRSMAS, preprocessed with the deblurring method, in another class. We have followed the same scheme: first, we have chosen the best model without data augmentation, and then we have evaluated several data augmentation techniques using the best model.

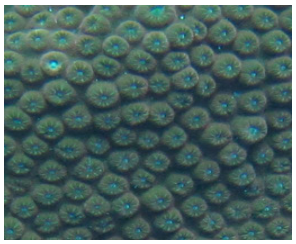
The results without data augmentation, along with the best hyperparameters, can be found in Table 10.10. We can see that all the models provide good results distinguishing textures and structures. This is actually the expected output as the images in both classes are very dissimilar. In this case the best model is ResNet-152, achieving an accuracy of 99.730%.

We have evaluated ResNet-152 using different data augmentation techniques. The ones that have given the best results can be seen in Table 10.11. The improvement is small as we already had a very good result. Even so, we have obtained an accuracy of 99.820%. This means that the model is only wrong in two images that can be seen in Figure 10.6. The first one (a) is a structure image classified as a texture image, and the second one (b) is a texture classified as a structure.

Table 10.11: Best results obtained for the texture or structure binary classifier using ResNet-152 and data augmentation techniques. The best accuracy is stressed in bold.

	shift = 0.4	zoom = 0.3	rotation = 4	flip
Accuracy	99.640	**99.820**	99.460	99.550

(a) MMEA



(b) MCAV

Figure 10.6: Coral images misclassified by the texture or structure binary classifier.

The high classification accuracy obtained by this binary classifier opens up the possibility of using it as a first level model to decide whether the image is a texture or a structure without losing accuracy in the second level.

10.6.5 *Two-level classifier: identification of coral species based on texture or structure images*

At this point, we already have the three models needed to build the two-level classifier for the classification of coral species based on either texture images or structure images. The first level of the classifier is to use the model that distinguishes between textures and structures, which we built in the previous section, to classify the input image in a texture image or a structure image. The second level of the two-level classifier is to classify the image into one of the coral species. To do this, the image is given to the RSMAS classifier or the StructureRSMAS classifier depending on the output of the first level.

We have evaluated the two-level classifier individually on structure images, on texture images and on structure images and texture images. The results can be seen in Table 10.12. It is important to note that, in each partition of the 5 fold cross validation, we are testing the two-level classifier with the same images that we used to test the StructureRSMAS model, the RSMAS model and the texture or structure binary

Table 10.12: Results obtained using the two-level classifier over the test set from RSMAS ∪ StructureRSMAS, RSMAS and StructureRSMAS.

	RSMAS ∪ StructureRSMAS	RSMAS	StructureRSMAS
Accuracy	93.874	98.356	85.263

classifier. That is, to test the two-level classifier we are not using images that were used to train any of its components.

Thanks to the high accuracy of the classifier in the first level, the accuracies obtained by the two-level classifier when we evaluate it only with texture images and only with structure images are very similar to the ones obtained by the RSMAS classifier and the StructureRSMAS classifier, respectively. This means that we classify textures and structures separately without decreasing the classification accuracy obtained by the texture classifier alone and the structure classifier alone. In fact, the accuracy for StructureRSMAS is now slightly better, which means that the RSMAS classifier in the second level is classifying correctly the image that are misclassified in the first level as a texture.

When we test the two-level classifier with texture and structure images, the accuracy is 93.874%, which is higher than the weighted arithmetic mean, taking into account the number of images in each dataset, between the obtained accuracy using the RSMAS model and the obtained accuracy using the StructureRSMAS model, which is 93.707%. Therefore, we have built a robust classifier that obtains a very good accuracy classifying any type of coral image.

10.7 CONCLUSIONS

The problem of classifying together structure and texture underwater coral images is a complicated task for three reasons: 1) the underwater images involve lightning problems and camera focusing problems due to the water, along with partial occlusion of marine animals; 2) different coral species look very similar and some species coexist together; and 3) there are not available coral structure datasets, and coral texture and structure images are very different from each other, no matter that they belong to the same class.

We have resolved this last problem by creating a coral structure dataset called StructureRSMAS, and we have tackled the classification of any coral image by using

CNNs. Particularly, we have used one of the newest and most powerful CNNs, ResNet, which have been the one that have obtained better accuracies in the texture dataset, RSMAS, and the structure dataset, StructureRSMAS. We have also used image enhancement methods in the two datasets, data augmentation techniques and transfer learning from ImageNet to improve our results.We have resolved this classification building a two-level classifier composed of three models: the best model known model for RSMAS, the best model we have developed in this work for StructureRSMAS and a model to distinguish between a texture image and a structure image, also developed in this work.

We have observed that data augmentation does not bring much benefit when classifying structure images alone, and we argue that this happens because StructureRSMAS is small.

The two-level classifier we have developed in this work first identifies if the input image is a texture or a structure and then uses one of two specialized CNNs, depending on whether the image is a texture or a structure. It is able to correctly classify 93.874%of the images in RSMAS ∪ StructureRSMAS.

ACKNOWLEDGMENTS

This work was partially supported by the Spanish Ministry of Science and Technology under the project TIN2017-89517-P. Siham Tabik was supported by the Ramón y Cajal Program (RYC-2015-18136). Anabel Gómez-Ríos was supported by the FPU Program 998758-2016. The NVIDIA Titan Xp used for this research was donated by the NVIDIA Corporation.

REFERENCES

[AD18a] N. Ani Brown Mary and D. Dejey, 'Coral reef image/video classification employing novel octa-angled pattern for triangular sub region and pulse coupled convolutional neural network (pccnn)', *Multimedia Tools and Applications*, Jun. 2018.

[AD18b] N. Ani Brown Mary and D. Dejey, 'Classification of coral reef submarine images and videos using a novel z with tilted z local binary pattern', *Wireless Personal Communications*, vol. 98, no. 3, pp. 2427–2459, Feb. 2018.

[AD17] N. Ani Brown Mary and D. Dharma, 'Coral reef image classification employing improved ldp for feature extraction', *Journal of Visual Communication and Image Representation*, vol. 49, pp. 225–242, 2017.

[BBA16] E. Basaeed, H. Bhaskar and M. Al-Mualla, 'Supervised remote sensing image segmentation using boosted convolutional neural networks', *Knowledge-Based Systems*, vol. 99, pp. 19–27, 2016.

[BEK+12] O. Beijbom, P. J. Edmunds, D. I. Kline, B. G. Mitchell and D. Kriegman, 'Automated annotation of coral reef survey images', in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 1170–1177.

[Cho+15] F. Chollet *et al.*, *Keras*, Accessed on 23-01-2019, 2015.

[CUW] *Coralpedia of the univerisity of warwick*, http://coralpedia.bio.warwick.ac.uk/en, Accessed on 23-01-2019.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, 'Imagenet: A large-scale hierarchical image database', in *Computer Vision and Pattern Recognition (CVPR), 2009. IEEE Conference on*, IEEE, 2009, pp. 248–255.

[Ela15] M. Elawady, 'Sparse coral classification using deep convolutional neural networks', *arXiv preprint arXiv:1511.09067*, 2015.

[EOL] *Encyclopedia of life*, http://eol.org/, Accessed on 23-01-2019.

[GTL+19] A. Gómez-Ríos, S. Tabik, J. Luengo, A. Shihavuddin, B. Krawczyk and F. Herrera, 'Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation', *Expert Systems with Applications*, vol. 118, pp. 315–328, 2019.

[GBC16] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[HZR+16] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[HLV+17] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, 'Densely connected convolutional networks', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 2261–2269.

[IUCNa] *Iucn red list of threatened species*, http://www.iucnredlist.org/, Accessed on 23-01-2019.

[IUCNb] *Iucn red list table of number of threatened species by major groups of organisms*, Available from: http://cmsdocs.s3.amazonaws.com/summarystats/2017-3_Summary_Stats_Page_Documents/2017_3_RL_Stats_Table_1.pdf, Accessed on 22-01-2019.

[KSH12] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[LBB+98] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[LYJ+17] G. Liu, Z. Yin, Y. Jia and Y. Xie, 'Passenger flow estimation based on convolutional neural network in public transportation system', *Knowledge-Based Systems*, vol. 123, pp. 102–115, 2017.

[MBA+16a] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. Fisher, 'Automatic annotation of coral reefs using deep learning', in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–5.

[MBA+16b] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. Fisher, 'Coral classification with hybrid feature representations', in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 519–523.

[MBA+17] A. Mahmood, M. Bennamoun, S. An, F. Sohel, F. Boussaid, R. Hovey, G. Kendrick and R. B. Fisher, 'Chapter 21 - deep learning for coral classification', in *Handbook of Neural Computation*, P. Samui, S. Sekhar and V. E. Balas, Eds., Academic Press, 2017, pp. 383–401.

[MSS05] M. S. A. C. Marcos, M. N. Soriano and C. A. Saloma, 'Classification of coral reef images from underwater video using neural networks', *Optics express*, vol. 13, no. 22, pp. 8766–8771, 2005.

[MAP+15] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[PRJ+08] O. Pizarro, P. Rigby, M. Johnson-Roberson, S. B. Williams and J. Colquhoun, 'Towards image-based marine habitat classification', in *OCEANS 2008*, IEEE, 2008, pp. 1–7.

[PAA+87] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman and K. Zuiderveld, 'Adaptive histogram equalization and its variations', *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[RDS+15] O. Russakovsky *et al.*, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[SP16] J. H. Seinfeld and S. N. Pandis, *Atmospheric chemistry and physics: from air pollution to climate change*. John Wiley & Sons, 2016.

[SGG+13] A. Shihavuddin, N. Gracias, R. Garcia, A. C. Gleason and B. Gintert, 'Image-based coral reef classification and thematic mapping', *Remote Sensing*, vol. 5, no. 4, pp. 1809–1841, 2013.

[Shi17] A. Shihavuddin, *Coral reef dataset, v2*. Mendeley data https://data.mendeley.com/datasets/86y667257h/2, Accessed on 12-02-2018, 2017.

[SZ14] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556*, 2014.

[SDB+14] J. T. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, 'Striving for simplicity: The all convolutional net', *arXiv preprint arXiv:1412.6806*, 2014.

[SD09] M. D. Stokes and G. B. Deane, 'Automated processing of coral reef benthic images', *Limnol. Oceanogr.: Methods*, vol. 7, no. 157, pp. 157–168, 2009.

[SLJ+15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, 'Going deeper with convolutions', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[SVI+16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, 'Rethinking the inception architecture for computer vision', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[TXL+18] H. Tang, B. Xiao, W. Li and G. Wang, 'Pixel convolutional neural network for multi-focus image fusion', *Information Sciences*, vol. 433, pp. 125–141, 2018.

[TGS+18] X. Tao, H. Gao, X. Shen, J. Wang and J. Jia, 'Scale-recurrent network for deep image deblurring', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8174–8182.

[XTJ+18] B. Xiao, H. Tang, Y. Jiang, W. Li and G. Wang, 'Brightness and contrast controllable image enhancement based on histogram specification', *Neurocomputing*, vol. 275, pp. 2798–2809, 2018.

[XWB+18] B. Xiao, K. Wang, X. Bi, W. Li and J. Han, '2d-lbp: An enhanced local binary feature for texture image classification', *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[Yu17] F. Yu, *Resnet and densenet cnns in keras*, https://github.com/flyyufelix/cnn_finetune, Accessed on 23-01-2019, 2017.

[ZMH16] D. Zhang, D. Meng and J. Han, 'Co-saliency detection via a self-paced multiple-instance learning framework', *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 5, pp. 865–878, 2016.

11

COVIDGR DATASET AND COVID-SDNET METHODOLOGY FOR PREDICTING COVID-19 BASED ON CHEST X-RAY IMAGES

Tabik, S., Gómez-Ríos, A., Martín-Rodríguez, J. L., Sevillano-García, I., Rey-Area, M., Charte, D., Guirado, E., Suárez, J. L., Luengo, J., Valero-González, M. A., García-Villanova, P., Olmedo-Sánchez, E. & Herrera, F. (2020). COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on chest X-ray images. *IEEE journal of biomedical and health informatics*, 24(12), 3595-3605.
DOI: https://doi.org/10.1109/JBHI.2020.3037127

- Status: Published
- Impact Factor (JCR 2020): 5.772
- Subject Category: Computer Science, Information Systems. Ranking 28/161 (Q1)
- Subject Category: Computer Science, Interdisciplinary Applications. Ranking 17/111 (Q1)
- Subject Category: Mathematical And Computational Biology. Ranking 5/58 (Q1)
- Subject Category: Medical Informatics. Ranking 4/30 (Q1)

COVIDGR DATASET AND COVID-SDNET METHODOLOGY FOR PREDICTING COVID-19 BASED ON CHEST X-RAY IMAGES

Siham Tabik[a], Anabel Gómez-Ríos[a], José Luis Martín-Rodríguez[b], Iván Sevillano-García[a], Manuel Rey-Area[c], David Charte[a], Emilio Guirado[d], Juan Luis Suárez[a], Julián Luengo[a], María Ángeles García-Villanova[b], Paloma García-Villanova[b], Eulalia Olmedo-Sánchez[b], Francisco Herrera[a]

[a] Andalusian Research Institute in Data Science and Computational Intelligence, Dept. of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
[b] Hospital Universitario Clínico San Cecilio de Granada, Spain
[c] atlanTTic Research Center for Telecommunication Technologies, University of Vigo, Galicia, Spain
[d] Multidisciplinary Institute for Environment Studies "Ramón Margalef", University of Alicante, Spain

ABSTRACT

Currently, Coronavirus disease (COVID-19), one of the most infectious diseases in the 21st century, is diagnosed using RT-PCR testing, CT scans and/or Chest X-Ray (CXR) images. CT (Computed Tomography) scanners and RT-PCR testing are not available in most medical centers and hence in many cases CXR images become the most time/cost effective tool for assisting clinicians in making decisions. Deep learning neural networks have a great potential for building COVID-19 triage systems and detecting COVID-19 patients, especially patients with low severity. Unfortunately, current databases do not allow building such systems as they are highly heterogeneous and biased towards severe cases. This paper is three-fold: (i) we demystify the high sensitivities achieved by most recent COVID-19 classification models, (ii) under a close collaboration with Hospital Universitario Clínico San Cecilio, Granada, Spain, we built COVIDGR-1.0, a homogeneous and balanced database that includes all levels of severity, from normal with Positive RT-PCR, Mild, Moderate to Severe. COVIDGR-1.0 contains 426 positive and 426 negative PA (PosteroAnterior) CXR views and (iii) we propose COVID Smart Data based Network (COVID-SDNet) methodology for improving the generalization capacity of COVID-classification models.

Our approach reaches good and stable results with an accuracy of 97.72% ± 0.95%, 86.90% ± 3.20%, 61.80% ± 5.49% in severe, moderate and mild COVID-19 severity levels. Our approach could help in the early detection of COVID-19. COVIDGR-1.0 along with the severity level labels are available to the scientific community through this link https://dasci.es/es/transferencia/open-data/covidgr/.

Keywords: *COVID-19*, *Smart Data*, *Convolutional Neural Networks*

11.1 INTRODUCTION

In the last months, the world has been witnessing how COVID-19 pandemic is increasingly infecting a large mass of people very fast everywhere in the world. The trends are not clear yet but some research confirm that this problem may persist until 2024 [KTG+20]. Besides, prevalence studies conducted in several countries reveal that a tiny proportion of the population have developed antibodies after exposure to the virus, e.g., 5% in Spain [1]. This means that frequently a large number of patients will need to be assessed in small time intervals by few number of clinicians and with very few resources.

In general, COVID-19 diagnosis is carried out using at least one of these three tests.

- Computed Tomography (CT) scans-based assessment: it consists in analyzing 3D radiographic images from different angles. The needed equipment for this assessment is not available in most hospitals and it takes more than 15 minutes per patient in addition to the time required for CT decontamination [2].
- Reverse Transcription Polymerase Chain Reaction (RT-PCR) test: it detects the viral RNA from sputum or nasopharyngeal swab [WLF+20]. It requires specific material and equipment, which are not easily accessible and it takes at least 12 hours, which is not desirable as positive COVID-19 patients should be identified and tracked as soon as possible. Some studies found that RT-PCR results from several tests at different points from the same patients were variable during the course of the illness producing a high false-negative rate [LYL+20]. The authors

1 https://english.elpais.com/society/2020-05-14/antibody-study-shows-just-5-of-spaniards-have-contracted-the-coronavirus.html

2 //www.acr.org/Advocacy-and-Economics/ACR-Position-Statements/Recommendations-for-Chest-Radiography-and-CT-for-Suspected-COVID19-Infection

suggested that RT-PCR test should be combined with other clinical tests such as CT.

- Chest X-Ray (CXR): The required equipment for this assessment are less cumbersome and can be lightweight and transportable. In general, this type of resources is more available than the required for RT-PCR and CT-scan tests. In addition, CXR test takes about 15 seconds per patient [WLF+20], which makes CXR one of the most time/cost effective assessment tools.

Few recent studies provide estimates on expert radiologists sensitivity in the diagnosis of COVID-19 based on CT scans, RT-PCR and CXR. A study on a set of 51 patients with chest CT and RT-PCR essay performed within 3 days, reported a sensitivity in CT of 98% compared with RT-PCR sensitivity of 71% [FZX+20]. A different study on 64 patients (26 men, mean age 56 ± 19 years) reported a sensitivity of 69% for CXR compared with 91% for initial RT-PCR [WLF+20]. According to an analysis of 636 ambulatory patients [WER+20], most patients presenting to urgent care centers with confirmed coronavirus disease 2019 have normal or mildly abnormal findings on CXR. Only 58.3% of these patients are correctly diagnosed by the expert eye.

In a recent study [WLF+20], authors proposed simplifying the quantification of the level of severity by adapting a previously defined Radiographic Assessment of Lung Edema (RALE) score [WZK+18] to COVID-19. This new score is calculated by assigning a value between 0-4 to each lung depending on the extent of visual features such as, consolidation and ground glass opacities, in the four parts of each lung as depicted in Fig. 11.1. Based on this score, experts can identify the level of severity of the infection among four severity stages, Normal 0, Mild 1-2, Moderate 3-5 and Severe 6-8. In practice, a patient classified by expert radiologist as Normal can have positive RT-PCR. We refer to these cases as Normal-PCR+. Expert annotation adopted in this work is based in this score.

Automated image analysis via Deep learning (DL) models have a great potential to optimize the role of CXR images for a fast diagnosis of COVID-19. A robust and accurate DL model could serve as a triage method and as a support for medical decision making. An increasing number of recent works claim achieving impressive sensitivities $> 95\%$, far higher than expert radiologists. These high sensitivities are due to the bias in the most used COVID-19 dataset, *COVID-19 Image Data Collection* [CMD20]. This dataset includes a very small number of COVID-19 positive cases, coming from highly heterogeneous sources (at least 15 countries) and most cases

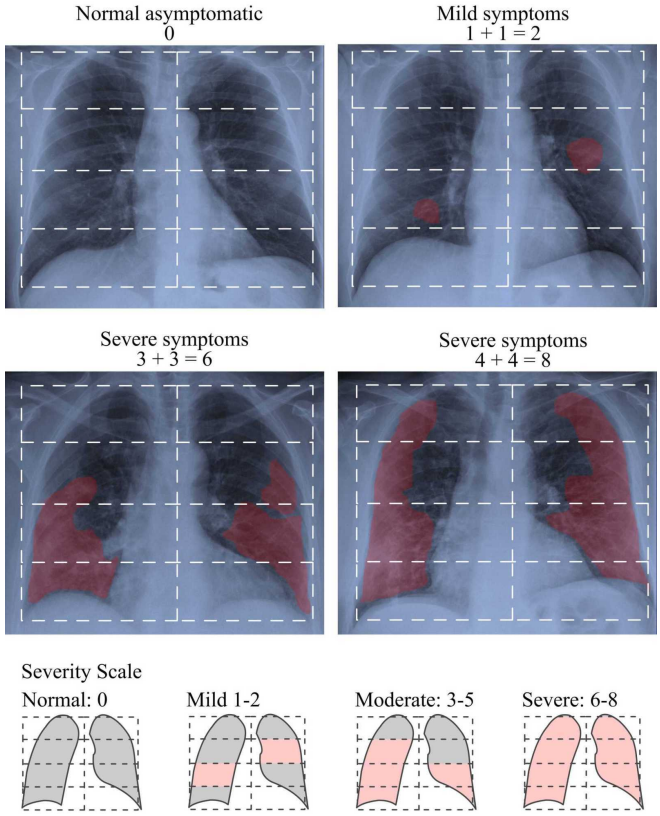


Figure 11.1: The stratification of radiological severity of COVID-19. Examples of how RALE index is calculated.

are severe patients, an issue that drastically reduces its clinical value. To populate Non-COVID and Healthy classes, AI researchers are using CXR images from diverse pulmonary disease repositories. The obtained models will have no clinical value as well since they will be unable to detect patients with low and moderate severity, which are the target of a clinical triage system. In view of this situation, there is still a huge need for higher quality datasets built under the same clinical protocol and under a close collaboration with expert radiologists.

Multiple studies have proven that higher quality data ensures higher quality models. The concept of Smart Data refers to the process of converting raw data into higher quality data with higher concentration of useful information [LGR+20]. Smart data includes all pre-processing methods that improve value and veracity of data. Examples of these methods include noise elimination, data-augmentation [TPH+17] and data transformation [RGT+20] among other techniques.

In this work, we designed a high clinical quality dataset, named COVIDGR-1.0 that includes four levels of severity, Normal-PCR+, Mild, Moderate and Severe. We identified these four severity levels from a recent COVID-19 radiological study [WLF+20]. We also propose COVID Smart Data based Network (COVID-SDNet) methodology. It combines segmentation, data-augmentation and data transformations together with an appropriate Convolutional Neural Network (CNN) for inference.

The contributions of this paper can be summarized as follows:

- We analyze reliability, potential and limitations of the most used COVID-19 CXR datasets and models.
- From a data perspective, we provide the first public dataset, called COVIDGR-1.0, that quantifies COVID-19 in terms of severity levels, normal, mild, moderate and severe, with the aim of building triage systems with high clinical value.
- From a pre-processing perspective, we combined several methods. To eliminate irrelevant information from the input CXR images, we used a new pre-processing method called segmentation-based cropping. To increase discrimination capacity of the classification model, we used a class-inherent transformation method inspired by GANs.
- From a post-processing perspective, we proposed a new inference process that fuses the predictions of the four transformed classes obtained by the class-inherent transformation method to calculate the final prediction.

- From a global perspective, we designed a novel methodology, named COVID-SDNet, with a high generalization capacity for COVID-19 classification based on CXR images. COVID-SDNet combines segmentation, data-transformation, data-augmentation, and a suitable CNN model together with an inference approach to get the final prediction.

Experiments demonstrate that our approach reaches good and stable results especially in moderate and severe levels, with $97.72\% \pm 0.95\%$ and $86.90\% \pm 3.20\%$ respectively. Lower accuracies were obtained in mild and normal-PCR+ severity levels with $61.80\% \pm 5.49\%$ and $28.42\% \pm 2.58\%$, respectively.

This paper is organized as follows: A review of the most used datasets and COVID-19 classification approaches is provided in Section 11.2. Section 11.3 describes how COVIDGR-1.0 is built and organized. Our approach is presented in Section 11.4. Experiments, comparisons and results are provided in Section 11.5. The inspection of the model's decision using heatmaps is provided in Section 11.6 and the conclusions are pointed out in Section 12.9.

11.2 RELATED WORKS

The last months have known an increasing number of works exploring the potential of deep learning models for automating COVID-19 diagnosis based on CXR images. The results are promising but still too much work needs to be done at the level of data and models design. Given the potential bias in this type of problems, several studies include explication methods to their models. This section analyzes the advantages and limitations of current datasets an models for building automatic COVID-19 diagnosis systems with and without decision explication.

11.2.1 *Datasets*

There does not exist yet a high quality collection of CXR images for building COVID-19 diagnosis systems of high clinical value. Currently, the main source for COVID-19 class is *COVID-19 Image Data Collection* [CMD20]. It contains 76 positive and 26 negative PA views. These images were obtained from highly heterogeneous equipment from all around the world. Another example of COVID-19 dataset is Figure-1-COVID-19 Chest X-ray Dataset Initiative [Chu20]. To build Non-COVID

Table 11.1: A brief description of COVIDx dataset [CMD20] (only PA views are counted).

Version	Normal(healthy)	Pneumonia	COVID-19
1.0	1,583	4,273 (Bacterial+viral)	76
2.0	8,066	8,614	190

classes, most studies are using CXR from one or multiple public pulmonary disease data-sets. Examples of these repositories are:

- RSNA Pneumonia CXR challenge dataset on Kaggle [RSNA19].
- ChestX-ray8 dataset [WPL+17].
- MIMIC-CXR dataset [JPG+19].
- PadChest dataset [BPS+20].

For instance, COVIDx 1.0 [WW20a] was built by combining three public datasets: (i) *COVID-19 Image Data Collection* [CMD20], (ii) Figure-1-COVID- 19 Chest X-ray Dataset Initiative [Chu20] and (iii) RSNA Pneumonia Detection Challenge dataset [RSNA19]. COVIDx 2.0 was built by re-organizing COVIDx 1.0 into three classes, Normal (healthy), Pneumonia and COVID-19, using 201 CXR images for COVID class, including PA(PosteroAnterior) and AP(AnteroPosterior) views (see Table 11.1). Notice that for a correct learning front view (PA) and back view (AP) cannot be mixed in the same class.

Although the value of these datasets is unquestionable as they are being useful for carrying out first studies and reformulations, they do not guarantee useful triage systems for the next reasons. It is not clear what annotation protocol has been followed for constructing the positive class in *COVID-19 Image Data Collection*. The included data is highly heterogeneous and hence DL-models can rely on other aspects than COVID visual features to differentiate between the involved classes. This dataset does not provide a representative spectrum of COVID-19 severity levels, most positive cases are of severe patients [KEG+20]. In addition, an interesting critical analysis of these datasets has shown that CNN models obtain similar results with and without eliminating most of the lungs in the input X-Ray images [MN20], which confirms again that there is a huge need of COVID-19 datasets with high clinical value.

Table 11.2: Summary of related works that analyze variations of COVIDx with CNN.

Ref.	Classes	Datasets	Model	Partition	Sens.	Acc.
[WW20a]	Normal, Pneumonia, COVID	COVIDx 1.0	COVIDNet	98% - 2%	87.1%	92.6%
[AHN+20]	Normal, COVID	COVIDx 1.0	COVID-CAPS	98% - 2%	90%	95.7%
[OTY+20]	No-Findings, COVID	[CMD20] + [WPL+17]	DarkCovidNet	5-FCV	90.65%	98.08%
	No-Findings, Pneumonia, COVID			5-FCV	97.9%	87.02%
[KDR+20]	Normal, Pneumonia, COVID	COVIDx 2.0 + [RSNA19]	VGG-19 + DenseNet-161	70% - 30%	93%	96.77%
[GT20]	Normal, Bacterial, Viral, COVID	[CMD20] + [RSNA19]	Bayesian ResNet50V2	80% - 20%	85.71%	89.82%
[AM20]	Normal, Pneumonia, COVID	[CMD20] + [RSNA19] + other sources	MobileNet	10-FCV	98.66%	96.78%

Table 11.3: A brief summary of COVIDGR-1.0 dataset. All samples in COVIDGR 1.0 are segmented CXR images considering only PA view.

Dataset	Class	#images	women	men	#img. per severity level
COVIDGR-1.0	Negative	426	239	187	
	COVID-19	426	190	236	Normal-PCR+: 76 Mild: 100 Moderate: 171 Severe: 79

Our claim is that the design of a high quality dataset must be done under a close collaboration between expert radiologists and AI experts. The annotations must follow the same protocol and representative numbers of all levels of severity, especially Mild and Moderate levels, must be included.

11.2.2 *DL classification models*

Existing related works are not directly comparable as they consider different combinations of public data-sets and different experimental setup. A brief summary of these works is provided in Table 11.2.

The most related studies to ours as they proposed different models to the typical ones are [WW20a] and [AHN+20]. In [WW20a], the authors designed a deep network, called COVIDNet. They affirmed that COVIDNet reaches an overall accuracy of 92.6%, with 97.0% sensitivity in Normal class, 90.0% in Non-COVID-19 and 87.1% in COVID-19. The authors of a smaller network, called COVID-CAPS [AHN+20], also claim that their model achieved an accuracy of 98.7%, sensitivity of 90%, and specificity of 95.8%. These results look too impressive when compared to expert radiologist sensitivity, 69%. This can be explained by the fact that the used dataset is biased to severe COVID cases [KEG+20]. In addition, the performed experiments in both cited works are not statistically reliable as they were evaluated on one single partition. The stability of these models, in terms of standard deviation, has not been reported.

11.2.3 *DL classification models with explanation approaches*

Several interesting explanations were proposed to help inspect the predictions of DL-models ([GT20; KDR+20]) although all their classification models were trained and validated on variations of COVIDx. The authors in [KDR+20] first use an ensemble of two CNN networks to predict the class of the input image, as Normal, Pneumonia or COVID. Then highlight class-discriminating regions in the input CXR image using gradient-guided class activation maps (Grad-CAM++) and layer-wise relevance propagation (LRP). In [GT20], the authors proposed explaining the decision of the classification model to radiologists using different saliency map types together with uncertainty estimations (i.e., how certain is the model in the prediction).

11.3 COVIDGR-1.0: DATA ACQUISITION, ANNOTATION AND ORGANIZATION

Instead of starting with an extremely large and noisy dataset, one can build a small and smart dataset then augment it in a way it increases the performance of the model. This approach has proven effective in multiple studies. This is particularly true in the medical field, where access to data is heavily protected due to privacy concerns and costly expert annotation.

Under a close collaboration with four highly trained radiologists from Hospital Universitario Clínico San Cecilio, Granada, Spain, we first established a protocol on how CXR images are selected and annotated to be included in the dataset. A

CXR image is annotated as COVID-19 positive if both RT-PCR test and expert radiologist confirm that decision within less than 24 hours. CXR with positive PCR that were annotated by expert radiologists as Normal are labeled as Normal-PCR+. The involved radiologists annotated the level of severity of positive cases based on RALE score as: Normal-PCR+, Mild, Moderate and Severe.

COVIDGR-1.0 is organized into two classes, positive and negative. It contains 852 images distributed into 426 positive and 426 negative cases, more details are provided in Table 11.3. All the images were obtained from the same equipment and under the same X-ray regime. Only PosteriorAnterior (PA) view is considered. COVIDGR-1.0 along with the severity level labels are available to the scientific community through this link: https://dasci.es/es/transferencia/open-data/covidgr/.

11.4 COVID-SDNET METHODOLOGY

In this section, we describe COVID-SDNet methodology in detail, covering pre-processing to produce smart data, including segmentation and data transformation for increasing discrimination between positive and negative classes, combined with a deep CNN for classification.

One of the pieces of COVID-SDNet is the CNN-based classifier. We have selected Resnet-50 initialized with ImageNet weights for a transfer learning approach. To adapt this CNN to our problem, we have removed the last layer of the net and added a 512 neurons layer with ReLU activation and a two or four neurons layer (according to the considered number of classes) with softmax activation.

Let X be the set of n images and K the total number of classes. Each image $\mathbf{x}_i \in X$ has a true label y_i with $i = 1, 2, \ldots, n$. The softmax function computes the probability that an image belongs to class k with $k = 1, \ldots, K$. Let $\mathbf{w} = (w_1, \ldots, w_K)$ be the output of the last fully connected layer before the softmax activation is applied. Then, this function is defined as: softmax : $\mathbb{R}^K \to [0,1]^K$,

$$\text{softmax}(\mathbf{w})_j = \frac{\exp(w_j)}{\sum_{k=1}^{K} \exp(w_k)} \ .$$

Let $\widehat{y_i}$ be the class prediction of the network for the image $\mathbf{x}_i$, then $\widehat{y_i} = \text{argmax}(\text{softmax}(\mathbf{w}))$, where $\mathbf{w}$ is the output vector of the last layer before softmax is applied for the input $\mathbf{x}_i$.

All the layers of the network were fine-tuned. We used a batch size of 16 and SGD as optimizer.

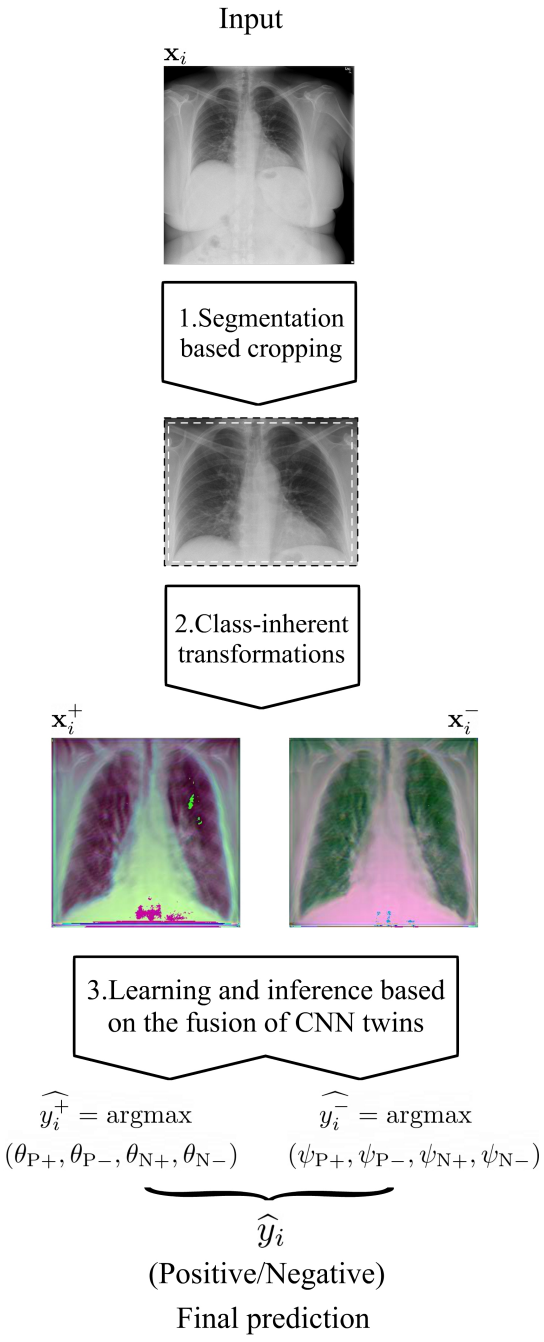


Figure 11.2: Flowchart of the proposed COVID-SDNet methodology.

(a) Input image

(b) The smallest rectangle that delimits the left and right segmented lungs is calculated

(c) 2.5% of pixels are added to the left, right, up and down sides of the rectangle then the final rectangle is cropped

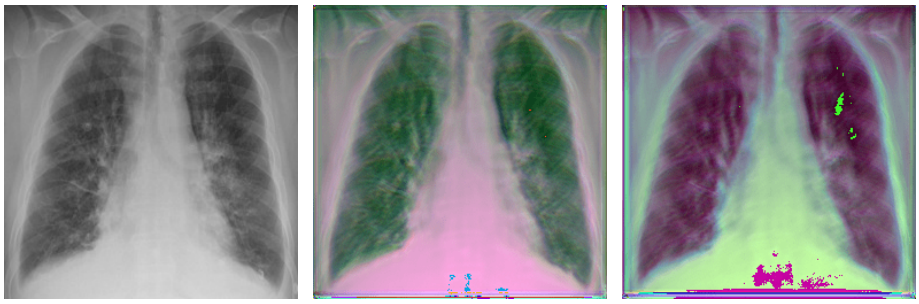
Figure 11.3: The segmentation-based cropping pre-processing applied to the input X-ray image

The main stages of COVID-SDNet are three, two associated to pre-processing for producing quality data (smart data stages) and the learning and inference process. A flowchart of COVID-SDNet is depicted in Fig. 11.2.

1. *Segmentation-based cropping: Unnecessary information elimination*

   Different CXR equipment brands include different extra information about the patient in the sides and contour of CXR images. The position and size of the patient may also imply the inclusion of more parts of the body, e.g., arms, neck, stomach. As this information may alter the learning of the classification model, first, we segment the lungs using the U-Net segmentation model provided in ([Min20]), pre-trained on Tuberculosis Chest X-ray Image datasets [JCA+14] and RSNA Pneumonia CXR challenge dataset [RSNA19]. Then, we calculate the smallest rectangle that delimits the left and right segmented-lungs. Finally, to avoid eliminating useful information, we add 2.5% of pixels to the left, right, up and down sides of the rectangle. The resulting rectangle is cropped. An illustration with example of this pre-processing is shown in Fig. 11.3.

2. *Class-inherent transformations Network*



(a) Original Negative (b) Negative transf. (c) Positive transf.

Figure 11.4: Class-inherent transformations applied to a negative sample. a) Original negative sample; b) Negative transformation; c) Positive transformation

To increase the discrimination capacity of the classification model, we used FuCiTNet [RGT+20], a Class-inherent transformations (CiT) Network inspired by GANs (Generative Adversarial Networks). This transformation method is actually an array of two generators G_{P} and G_{N}, where $_{\mathrm{P}}$ refers to the positive class and $_{\mathrm{N}}$ refers to the negative class. G_{P} learns the inherent-class transformations of the positive class P and G_{N} learns the inherent-class transformations of the negative class N. In other words, G_{P} learns the transformations that bring an input image from its own k domain, with $k \in \{\mathrm{P}, \mathrm{N}\}$, to the P class domain. Similarly, G_{N} learns the transformations that bring the input image from its k space, with $k \in \{\mathrm{P}, \mathrm{N}\}$, to the N class space. The classification loss is introduced in the generators to drive the learning of each specific k-class transformations. That is, each generator is optimized based on the following loss function:

$$\mathcal{L}_{gen_k} = l_{MSE} + 0.006 \cdot l_{Perceptual} + \lambda \cdot l_{CE}(\mathrm{y} == k) \tag{11.1}$$

Where l_{MSE} is a pixel-wise Mean Square Error, $l_{Perceptual}$ is a perception Mean Square Error and l_{CE} is the classifier loss. The weighted factor λ indicates how much the generator must change its outcome to suit the classifier. More details about these transformation networks can be found in [RGT+20].

The architecture of the generators consists of 5 identical residual blocks. Each block has two convolutional layers with 3×3 kernels and 64 feature maps followed by batch-normalization layers and Parametric ReLU as activation function. The last residual block is followed by a final convolutional layer which

reduces the output image channels to 3 to match the input's dimensions. The classifier is a ResNet-18 which consists of an initial convolutional layer with 7×7 kernels and 64 feature maps followed by a 3×3 max pool layer. Then, 4 blocks of two convolutional layers with 3×3 kernels with 64, 128, 256 and 512 feature maps respectively followed by a 7×7 average pooling and one fully connected layer which outputs a vector of N elements. ReLU is used as activation function.

Once the generators learn the corresponding transformations, the dataset is processed using G_{P} and G_{N}. Two pair of images $(\mathbf{x}_i^+, \mathbf{x}_i^-)$ will be obtained from each input image $\mathbf{x}_i$, $i = 1, \ldots, n$, where $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ are respectively the positively and negatively transformed images of $\mathbf{x}_i$. Note that, once the entire dataset is processed, we have four classes $(\mathrm{P}+, \mathrm{P}-, \mathrm{N}+, \mathrm{N}-)$ instead the original P and N classes. Let y_i be the class of $\mathbf{x}_i$, $y_i \in \{\mathrm{P}, \mathrm{N}\}$. If $y_i = \mathrm{P}$, G_{P} and G_{N} will produce the positive transformation $\mathbf{x}_i^+$ with $y_i^+ = \mathrm{P}+$ and the negative transformation $\mathbf{x}_i^-$ with $y_i^- = \mathrm{P}-$, respectively. If $y_i = \mathrm{N}$, G_{P} and G_{N} will produce the positive transformation $\mathbf{x}_i^+$ with $y_i^+ = \mathrm{N}+$ and the negative transformation $\mathbf{x}_i^-$ with $y_i^- = \mathrm{N}-$, respectively. Fig. 11.4 illustrates with example the transformations applied by G_{N} and G_{P}. Notice that these transformations are not meant to be interpretable by the human eye but rather help the classification model better distinguish between the different classes.

3. *Learning and inference based on the fusion of CNN twins*

The CNN classification model described above in this section (Resnet-50) is trained to predict the new four classes: $\mathrm{P}+, \mathrm{P}-, \mathrm{N}+, \mathrm{N}-$. The output of the network (after softmax is applied) for each transformed image associated to the original one is a vector $\boldsymbol{\theta} = (\theta_{\mathrm{P}+}, \theta_{\mathrm{P}-}, \theta_{\mathrm{N}+}, \theta_{\mathrm{N}-})$, where θ_j is the probability of the transformed image to belong to class $j \in \{\mathrm{P}+, \mathrm{P}-, \mathrm{N}+, \mathrm{N}-\}$. Herein, we propose an inference process to fuse the output of the two transformed images $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ to predict the label of the original image $\mathbf{x}_i$. In this way, for each pair $(\mathbf{x}_i^+, \mathbf{x}_i^-)$, the prediction of the original image $\widehat{y_i}$ will be either P or N. Let $\widehat{y_i^+} = \operatorname{argmax} \boldsymbol{\theta} = \operatorname{argmax} (\theta_{\mathrm{P}+}, \theta_{\mathrm{P}-}, \theta_{\mathrm{N}+}, \theta_{\mathrm{N}-})$ and $\widehat{y_i^-} = \operatorname{argmax} \boldsymbol{\psi} = \operatorname{argmax} (\psi_{\mathrm{P}+}, \psi_{\mathrm{P}-}, \psi_{\mathrm{N}+}, \psi_{\mathrm{N}-})$ be the ResNet-50 predictions for x_i^+ and x_i^- respectively. Then:

a) If $\widehat{y_i^+} = \mathrm{N}+$ and $\widehat{y_i^-} = \mathrm{N}-$, then $\widehat{y_i} = \mathrm{N}$.

b) If $\widehat{y_i^+}$ = P+ and $\widehat{y_i^-}$ = P−, then $\widehat{y_i}$ = P.

c) If none of the above applies, then

$$\widehat{y_i} = \begin{cases} \text{N if } \max(\theta_{Nj}, \psi_{Nj}) > \max(\theta_{Pj}, \psi_{Pj}), & j \in \{+, -\} \\ \text{P otherwise} \quad . \end{cases}$$

Experimentally, we used a batch size of 16 and SGD as optimizer.

11.5 EXPERIMENTS AND RESULTS

In this section we (1) provide all the information about the used experimental setup, (2) evaluate two state-of-the-art COVID classification models and FuCiTNet alone [RGT+20] on our dataset then, analyze (3) the impact of data pre-processing and (4) Normal-PCR+ severity level on our approach.

11.5.1 *Experimental setup*

Due to the high variations between different executions, we performed 5 different 5 fold cross validations in all the experiments. Each experiment uses 80% of COVIDGR-1.0 for training and the remaining 20% for testing. To choose when to stop the training process, we used a random 10% of each training set for validation. In each experiment, a proper set of data-augmentation techniques is carefully selected. All results, in terms of sensitivity, specificity, precision, F1 and accuracy, are presented using the average values and the standard deviation of the 25 executions. The used metrics are calculated as follows:

$$\text{recall(positive class)} = \textit{sensitivity} = \frac{\text{TP}}{\text{actual positives}}$$

$$\text{recall(negative class)} = \textit{specificity} = \frac{\text{TN}}{\text{actual negatives}}$$

$$\text{precision(positive class)} = \frac{\text{TP}}{\text{predicted positives}}$$

Table 11.4: COVIDNet and COVID-CAPS results on our dataset

Class	Negative		Positive (COVID-19)		Accuracy
Metric	Specificity	Precision	Sensitivity	Precision	
COVIDNet-CXR A [WW20a]	0.23	16.00	**99.29**	33.54	49.76
Retrained COVIDNet-CXR A	**88.82±0.90**	3.36±6.15	46.82±17.59	**81.65±6.02**	**67.82±6.11**
COVID-CAPS [AHN+20]	26.30	45.81	69.01	48.36	47.66
Retrained COVID-CAPS	65.74±9.93	**65.62±3.98**	64.93±9.71	66.07±4.49	65.34±3.26

$$\text{precision}(\text{negative class}) = \frac{\text{TN}}{\text{predicted negatives}}$$

$$\textit{accuracy} = \frac{\text{TP+TN}}{\text{total predictions}}$$

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

TP and TN refers respectively to the number of true positives and true negatives.

11.5.2 *Analysis of COVIDNet and COVID-CAPS*

We compare our approach with the two most related approaches to ours, COVIDNet [WW20a] and COVID-CAPS [AHN+20].

- COVIDNet: Currently, the authors of this network provide three versions, namely A, B and C, available at [WW20b]. A has the largest number of trainable parameters, followed by B and C. We performed two evaluations of each network in such a way that the results will be comparable to ours.
  - First, we tested COVIDNet-A, COVIDNet-B and COVIDNet-C, pre-trained on COVIDx, directly on our dataset by considering only two classes: Normal (negative), and COVID-19 (positive). The whole dataset (426 positive images and 426 negative images) is evaluated. We report in Table 11.4 recall and precision results for Normal and COVID-19 classes.

Table 11.5: Results of COVID-19 prediction using Retrained COVIDNet-CXR A, Retrained COVID-CAPS, ResNet-50 with and without segmentation, FuCiTNet and COVID-SDNet. All four levels of severity in the positive class are taken into account.

Class	N			P			Accuracy
Metric	**Specificity**	Precision	F1	**Sensitivity**	Precision	F1	
COVIDNet-CXR	88.82±0.90	3.36±6.15	73.31±3.79	46.82±17.59	81.65±6.02	56.94±15.05	67.82±6.11
COVID-CAPS	65.74±9.93	65.62±3.98	65.15±5.02	64.93±9.71	66.07±4.49	64.87±4.92	65.34±3.26
Without seg.	79.87±8.91	71.91±3.12	75.40±4.91	68.63±6.08	78.75±6.31	72.689±3.45	74.25±3.61
With seg.	78.41±7.09	73.36±4.66	75.46±2.97	70.80±8.26	77.17±4.79	73.40±4.01	74.60±2.93
FuCiTNet	**80.79±6.98**	72.00±4.48	75.84±3.18	67.90±8.58	78.48±4.99	72.35±4.76	74.35±3.34
COVID-SDNet	79.76±6.19	**74.74±3.89**	**76.94±2.82**	**72.59±6.77**	**78.67±4.70**	**75.71±3.35**	**76.18±2.70**

  - Second, we retrained COVIDNet on our dataset. It is important to note that as only a checkpoint of each model is available, we could not remove the last layer of these networks, which has three neurons. We used 5 different 5 fold cross validations. In order to be able to retrain COVIDNet models, we had to add a third Pneumonia class into our dataset. We randomly selected 426 images from the Pneumonia class in COVIDx dataset. We used the same hyper-parameters as the ones indicated in their training script, that is, 10 epochs, a batch size of 8 and a learning rate of 0.0002. We changed covid_weight to 1 and covid_percent to 0.33 since we had the same number of images in all the classes. Similarly, we report in Table 11.4 recall and precision of our two classes, Normal and COVID-19, and omit recall and precision of Pneumonia class. The accuracy reported in the same table only takes into account the images from our two classes. As with our models, we report here the mean and standard deviation of all metrics.

  Although we analyzed all three A, B and C variations of COVIDNet, for simplicity we only report the results of the best one.

- COVID-CAPS: This is a capsule network-based model proposed in [AHN+20]. Its architecture is notably smaller than COVIDNet, which implies a dramatically lower number of trainable parameters. Since the authors also provide a checkpoint with weights trained in the COVIDx dataset, we were able to follow a similar procedure than with COVIDNet:

- First, we tested the pretrained weights using COVIDx on COVIDGR-1.0 dataset. COVID-CAPS is designed to predict two classes, so we reused the same architecture with the new dataset and compute the evaluation metrics shown in Table 11.4.
- Second, COVID-CAPS architecture was retrained over the COVIDGR-1.0 dataset. This process finetunes the weights to improve class separation. The retraining process is performed using the same setup and hyper-parameters reported by the authors. Adam optimizer is used across 100 epochs with a batch size of 16. Class weights were omitted as with COVIDNet, since this dataset contains balanced classes in training as well as in test. Evaluation metrics are computed for five sets of 5-fold cross-validation test subsets and summarized in Table 11.4.

The results from Table 11.4 show that COVIDNet and COVID-CAPS trained on COVIDx overestimate COVID-19 class in our dataset, i.e., most images are classified as positive, resulting in very high sensitivities but at the cost of low positive predictive value. However, when COVIDNet and COVID-CAPS are re-trained on COVIDGR-1.0 they achieve slightly better overall accuracy and a higher balance between sensitivity and specificity, although they seem to acquire a bias favoring the negative class. In general, none of these models perform adequately for the detection of the disease from CXR images in our dataset.

11.5.3 *Results and Analysis of COVID prediction*

The results of the baseline COVID classification model considering all the levels of severity, with and without segmentation, FuCiTNet [RGT+20], and COVID-SDNet are shown in Table 11.5.

In general, COVID-SDNet achieves better and more stable results than the rest of approaches. In particular, COVID-SDNet achieved the highest balance between specificity and sensitivity with 76.94 ± 2.82 F1 in the negative class and 75.71 ± 3.35 F1 in the positive class. Most importantly, COVID-SDNet achieved the best sensitivity 72.59 ± 6.77 and accuracy with 76.18 ± 2.70. FuCiTNet provides in general good but lower and less stable results than COVID-SDNet. When comparing the results of the baseline classification model with and without segmentation, we can observe that the use of segmentation improves substantially the sensitivity, which is the

Table 11.6: Results of COVID-SDNet per severity level.

S (Severity level)	accuracy (S)(%)
Normal-PCR+	28.42 ± 2.58
Mild	61.80 ± 5.49
Moderate	86.90 ± 3.20
Severe	97.72 ± 0.95

most important criteria for a triage system. This can be explained by the fact that segmentation allows the model to focus on most important parts of the CXR image.

11.5.4 *Analysis per severity level*

To determine which levels are the hardest to distinguish by the best approach, we have analyzed the accuracy per severity level (S), with $\text{accuracy(S)} = \frac{\text{Correct predictions(S)}}{\text{Total number(S)}}$, where $S \in \{\text{Normal-PCR+, Mild, Moderate, Severe}\}$. The results are shown in Table 11.6.

As it can be seen from these results, COVID-SDNet correctly distinguish Moderate and Severe levels with an accuracy of 86.90% and 97.72%, respectively. This is due to the fact that Moderate and Severe CRX images contain more important visual features than Mild and Normal-PCR+ which ease the classification task. Normal-PCR+ and Mild cases are much more difficult to identify as they contain few or none visual features. These results are coherent with the clinical studies provided in [WER+20] and [WLF+20] which report that expert sensitivity is very low in Normal-PCR+ and Mild infection levels. Recall that the expert eye does not see any visual signs in Normal-PCR+ although the PCR is positive. Those cases are actually considered as asymptomatic patients.

11.5.5 *Analysis of the impact of Normal-PCR+*

To analyze the impact of Normal-PCR+ class on COVID-19 classification, we trained and evaluated the baseline model, FuciTNet, COVID-SDNet classification stage, COVIDNet-CXR-A and COVID-CAPS, on COVIDGR-1.0 by eliminating Normal-PCR+. The results are summarized in Table 11.7.

Table 11.7: Results of the baseline classification model with segmentation, COVID-SDNet, retrained COVIDNet-CXR-A and retrained COVID-CAPS. Only three levels of severity are considered, Mild, Moderate and Severe.

Class	N			P			Accuracy
Metric	**Specificity**	Precision	F1	**Sensitivity**	Precision	F1	
COVIDNet-CXR	83.42± 15.39	69.73± 10.34	74.45± 8.86	61.82± 22.44	79.50± 11.47	65.64± 15.90	72.62± 7.6
COVID-CAPS	65.09± 10.51	71.72± 5.57	67.52±5.29	73.31±9.74	68.40±5.13	70.20±4.31	69.20±3.61
With seg.	80.57±8.72	78.68±6.57	78.97±3.20	76.80±10.15	80.70±5.56	78.01±4.29	78.69±3.00
FuCiTNet	82.63±6.61	**79.94±4.28**	81.05±3.44	**78.91±5.88**	82.43±5.43	80.37±3.16	80.77±3.15
COVID-SDNet	**85.20±5.38**	78.88±3.89	**81.75±2.74**	76.80±6.30	**84.23±4.59**	**80.07±0.04**	**81.00±2.87**

Table 11.8: Results of COVID-SDNet by severity level without considering Normal-PCR+.

S (Severity level)	accuracy (S)(%)
Mild	46.00 ± 7.10
Moderate	85.38 ± 1.85
Severe	97.22 ± 1.86

Overall, all the approaches systematically provide better results when eliminating Normal-PCR+ from the training and test processes, including COVIDNet-CXR-A and COVID-CAPS. In particular, COVID-SDNet still represents the best and most stable approach.

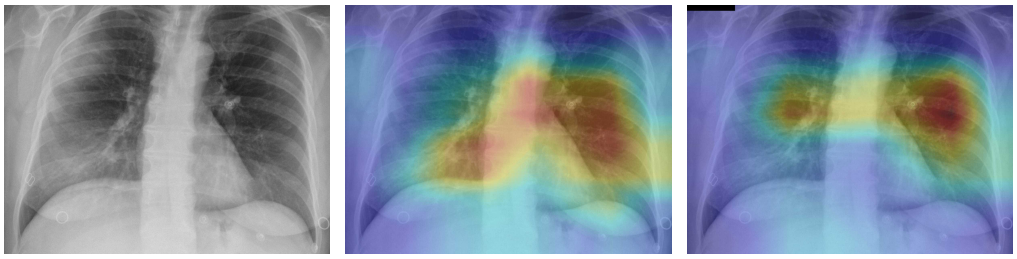
11.5.6 *Analysis per severity level*

A further analysis of the accuracy at the level of each severity degree (see Table 11.8) demonstrates that eliminating Normal-PCR+ decreases the accuracy in Mild and Moderate severity levels by 15.8% and 1.52% respectively.

These results show that although Normal-PCR+ is the hardest level to predict, its presence improves the accuracy of lower severity levels, especially Mild level.

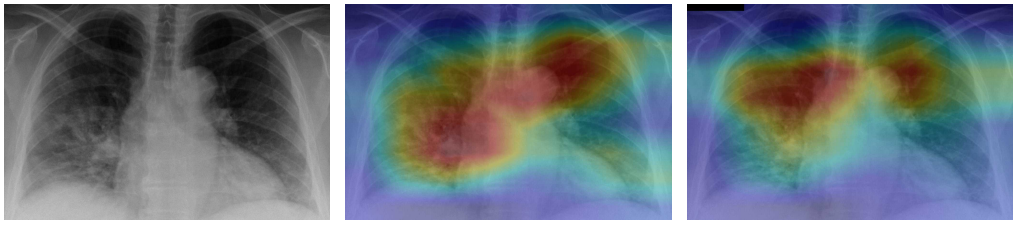
11.6 INSPECTION OF MODEL'S DECISION

Automatic DL diagnosis systems alone are not mature yet to replace expert radiologists. To help clinician making decisions, these tools must be interpretable so



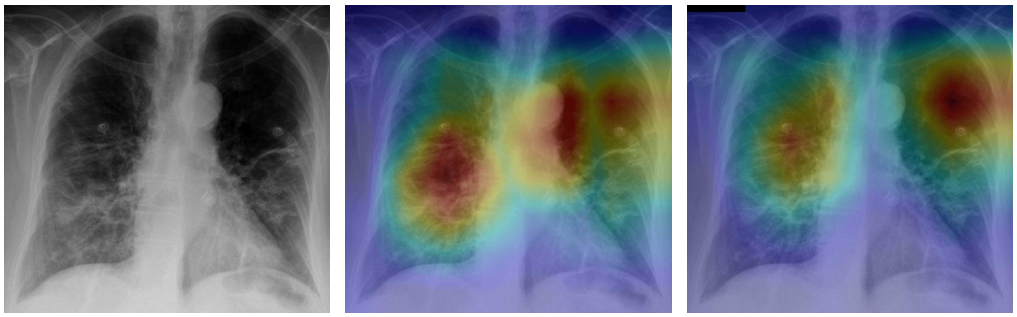
(a) Original Positive (Mild) (b) why positive (c) why negative

Figure 11.5: Heatmap showing the parts of the input image that triggered the positive prediction (b) and counterfactual explanation (c)



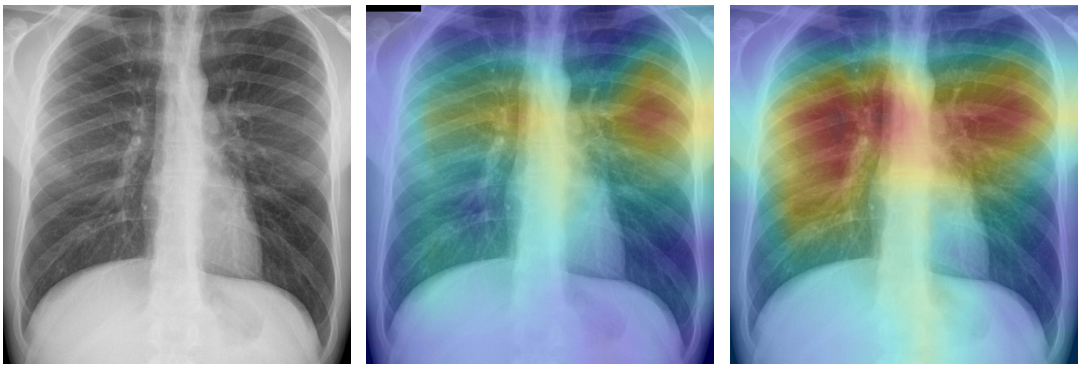
(a) Original Positive (Moderate) (b) why positive (c) why negative

Figure 11.6: Heatmap showing the parts of the input image that triggered the positive prediction (b) and counterfactual explanation (c)



(a) Original Positive (Severe) (b) why positive (c) why negative

Figure 11.7: Heatmap showing the parts of the input image that triggered the positive prediction (b) and counterfactual explanation (c)



(a) Original Negative (b) why positive (c) why negative

Figure 11.8: Heatmap that explains the parts of the input image that triggered the counterfactual explanation (b) and the negative actual prediction (c).

that clinicians can decide whether to trust the model or not [ADD+20]. We inspect what led our model make a decision by showing the regions of the input image that triggered that decision along with its counterfactual explanation by showing the parts that explain the opposite class. We adapted Grad-CAM method [SCD+17] to explain the decision of the negative and positive class.

Fig. 11.5, 11.6 and 11.7 show (a) the original CXR image, (b) visual explanation by means of a heat-map that highlights the regions/pixels which led the model to output the actual prediction and (c) its counterfactual explanation using a heat-map that highlights the regions/pixels which had the highest impact on predicting the opposite class. Higher intensity in the heat-map indicates higher importance of the corresponding pixel in the decision. The larger higher intensity areas in the heat-map determine the final class. However, Fig. 11.8(b) represents first the counterfactual explanation and Fig. 11.8(c) represents the explanation of the actual decision.

As expected, negative and positive interpretations are complementary, i.e, areas which triggered the correct decision are opposite, in most cases, to the areas that triggered the decision towards negative. In CXR images with different severity levels, the heat-maps correctly point out opaque regions due to different levels of infiltrates, consolidations and also to osteoarthritis.

In particular, in Fig. 11.5(b), the red areas in the right lung points out a region with infiltrates and also osteoarthritis in the spine region. Fig. 11.6 (b) correctly shows moderate infiltrates in the right lower and lower-middle lung fields in addition to a

dilation of ascending aorta and aortic arch (red color in the center). Fig. 11.5(c) shows normal upper-middle fields of both lungs (less important on the left due to aortic dilation). Fig. 11.7(b) indicates an important bilateral pulmonary involvement with consolidations.

As it can be observed in Fig. 11.8(c), the explanation of the negative class correctly highlights a symmetric bilateral pattern that occupies a larger lung volume especially in regions with high density. In fact, a very similar pattern is shown in the counterfactual explanation of the positive class in Fig. 11.5(c), 11.6(c) and 11.7(c).

11.7 CONCLUSIONS

This paper introduced a dataset, named COVIDGR-1.0, with high clinical value. COVIDGR-1.0 includes the four main COVID severity levels identified by a recent radiological study [WLF+20]. We proposed a methodology, called COVID-SDNet, that combines segmentation, data-augmentation and data transformation. The obtained results show the high generalization capacity of COVID-SDNet, specially on severe and moderate levels as they include important visual features. The existence of few or none visual features in Mild and Normal-PCR+ reduces the opportunities for improvement.

As main conclusions, we must highlight that COVID-SDNet can be used in a triage system to detect especially moderate and severe patients. Finally, we must also mention that more robust and accurate triage system can be built by fusing our approach with other approaches such as the one proposed in [CDR+20].

As future work, we are working on enriching COVIDGR-1.0 with more CXR images coming from different hospitals. We are planning to explore the use of additional clinical information along with CXR images to improve the prediction performance.

ACKNOWLEGMENTS

This work was supported by the project DeepSCOP-Ayudas Fundación BBVA a Equipos de Investigación Científica en Big Data 2018, COVID19_RX-Ayudas Fundación BBVA a Equipos de Investigación Científica SARS-CoV-2 y COVID-19 2020, and the Spanish Ministry of Science and Technology under the project TIN2017-89517-P. S. Tabik was supported by the Ramon y Cajal Programme (RYC-2015-18136). A. Gómez-Ríos was supported by the FPU Programme FPU16/04765. D. Charte was

supported by the FPU Programme FPU17/04069. J. Suárez was supported by the FPU Programme FPU18/05989. E.G was supported by the European Research Council (ERC Grant agreement 647038 [BIODESERT])

ETHICS

This project is approved by the Provincial Research Ethics Committee of Granada.

REFERENCES

[AHN+20] P. Afshar, S. Heidarian, F. Naderkhani, A. Oikonomou, K. N. Plataniotis and A. Mohammadi, 'Covid-caps: A capsule network-based framework for identification of covid-19 cases from x-ray images', *Pattern Recognition Letters*, vol. 138, pp. 638–643, 2020.

[AM20] I. Apostolopoulos and T. Mpesiana, 'Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks', *Physical and Engineering Sciences in Medicine*, p. 1, 2020.

[ADD+20] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, 'Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai', *Information fusion*, vol. 58, pp. 82–115, 2020.

[BPS+20] A. Bustos, A. Pertusa, J.-M. Salinas and M. de la Iglesia-Vayá, 'Padchest: A large chest x-ray image dataset with multi-label annotated reports', *Medical image analysis*, vol. 66, p. 101 797, 2020.

[Chu20] A. Chung, 'Figure 1 COVID-19 chest X-ray dataset initiative', 2020.

[CDR+20] J. P. Cohen, L. Dao, K. Roth, P. Morrison, Y. Bengio, A. F. Abbasi, B. Shen, H. K. Mahsa, M. Ghassemi, H. Li *et al.*, 'Predicting covid-19 pneumonia severity on chest x-ray with deep learning', *Cureus*, vol. 12, no. 7, 2020.

[CMD20] J. P. Cohen, P. Morrison and L. Dao, 'Covid-19 image data collection', *arXiv preprint arXiv:2003.11597*, 2020.

[FZX+20] Y. Fang, H. Zhang, J. Xie, M. Lin, L. Ying, P. Pang and W. Ji, 'Sensitivity of chest ct for covid-19: Comparison to rt-pcr', *Radiology*, vol. 296, no. 2, E115–E117, 2020.

[GT20] B. Ghoshal and A. Tucker, 'Estimating uncertainty and interpretability in deep learning for coronavirus (COVID-19) detection', *arXiv preprint arXiv:2003.10769*, 2020.

[JCA+14] S. Jaeger, S. Candemir, S. Antani, Y.-X. J. Wáng, P.-X. Lu and G. Thoma, 'Two public chest x-ray datasets for computer-aided screening of pulmonary diseases', *Quantitative imaging in medicine and surgery*, vol. 4, no. 6, p. 475, 2014.

[JPG+19] A. E. Johnson, T. J. Pollard, N. R. Greenbaum, M. P. Lungren, C.-y. Deng, Y. Peng, Z. Lu, R. G. Mark, S. J. Berkowitz and S. Horng, 'Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs', *arXiv preprint arXiv:1901.07042*, 2019.

[KDR+20] M. Karim, T. Döhmen, D. Rebholz-Schuhmann, S. Decker, M. Cochez, O. Beyan *et al.*, 'Deepcovidexplainer: Explainable covid-19 predictions based on chest x-ray images', *arXiv preprint arXiv:2004.04582*, 2020.

[KTG+20] S. M. Kissler, C. Tedijanto, E. Goldstein, Y. H. Grad and M. Lipsitch, 'Projecting the transmission dynamics of sars-cov-2 through the post-pandemic period', *Science*, vol. 368, no. 6493, pp. 860–868, 2020.

[KEG+20] S. Kundu, H. Elhalawani, J. W. Gichoya and C. E. Kahn Jr, *How might ai and chest imaging help unravel covid-19's mysteries?*, 2020.

[LYL+20] Y. Li, L. Yao, J. Li, L. Chen, Y. Song, Z. Cai and C. Yang, 'Stability issues of rt-pcr testing of sars-cov-2 for hospitalized patients clinically diagnosed with covid-19', *Journal of medical virology*, vol. 92, no. 7, pp. 903–908, 2020.

[LGR+20] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García and F. Herrera, 'Big data preprocessing', *Cham: Springer*, 2020.

[MN20] G. Maguolo and L. Nanni, 'A critic evaluation of methods for covid-19 automatic detection from x-ray images', *arXiv preprint arXiv:2004.12823*, 2020.

[Min20] E. Mineo, *U-Net lung segmentation*, Available at: https://www.kaggle.com/eduardomineo/u-net-lung-segmentation-montgomery-shenzhen, 2020.

[OTY+20] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim and U. R. Acharya, 'Automated detection of covid-19 cases using deep neural networks with x-ray images', *Computers in biology and medicine*, vol. 121, p. 103 792, 2020.

[RSNA19] *Radiological society of north america. RSNA pneumonia detection challenge*, https://www.kaggle.com/c/rsnapneumonia-detection-challenge/data, 2019.

[RGT+20] M. Rey-Area, E. Guirado, S. Tabik and J. Ruiz-Hidalgo, 'Fucitnet: Improving the generalization of deep learning networks by the fusion of learned class-inherent transformations', *Information Fusion*, vol. 63, pp. 188–195, 2020.

[SCD+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, 'Grad-cam: Visual explanations from deep networks via gradient-based localization', in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[TPH+17] S. Tabik, D. Peralta, A. Herrera-Poyatos and F. Herrera, 'A snapshot of image pre-processing for convolutional neural networks: Case study of mnist', 2017.

[WW20a] L. Wang and A. Wong, 'COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest radiography images', *arXiv preprint arXiv:2003.09871*, 2020.

[WW20b] L. Wang and A. Wong, *COVIDNet*, Available at: https://github.com/lindawangg/COVID-Net, 2020.

[WPL+17] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri and R. M. Summers, 'Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2097–2106.

[WZK+18] M. A. Warren, Z. Zhao, T. Koyama, J. A. Bastarache, C. M. Shaver, M. W. Semler, T. W. Rice, M. A. Matthay, C. S. Calfee and L. B. Ware, 'Severity scoring of lung oedema on the chest radiograph is associated with clinical outcomes in ards', *Thorax*, vol. 73, no. 9, pp. 840–846, 2018.

[WER+20] M. B. Weinstock, A. Echenique, J. Russell, A. Leib, J. Miller, D. Cohen, S. Waite, A. Frye and F. Illuzzi, 'Chest x-ray findings in 636 ambulatory patients with covid-19 presenting to an urgent care center: A normal chest x-ray is no guarantee', *J Urgent Care Med*, vol. 14, no. 7, pp. 13–18, 2020.

[WLF+20] H. Y. F. Wong, H. Y. S. Lam, A. H.-T. Fong, S. T. Leung, T. W.-Y. Chin, C. S. Y. Lo, M. M.-S. Lui, J. C. Y. Lee, K. W.-H. Chiu, T. W.-H. Chung *et al.*, 'Frequency and distribution of chest radiographic findings in patients positive for covid-19', *Radiology*, vol. 296, no. 2, E72–E78, 2020.

12

A ROBUST APPROACH FOR DEEP NEURAL NETWORKS IN PRESENCE OF LABEL NOISE: RELABELLING AND FILTERING INSTANCES DURING TRAINING

Gómez-Ríos, A., Luengo, J. & Herrera, F. (2022). A robust approach for deep neural networks in presence of label noise: relabelling and filtering instances during training.

- Status: Submitted to IEEE Transactions on Neural Networks and Learning Systems

A ROBUST APPROACH FOR DEEP NEURAL NETWORKS IN PRESENCE OF LABEL NOISE: RELABELLING AND FILTERING INSTANCES DURING TRAINING

Anabel Gómez-Ríos[a], Julián Luengo[a], Francisco Herrera[a]

[a] Andalusian Research Institute in Data Science and Computational Intelligence, Dept. of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain

ABSTRACT

Deep learning has outperformed other machine learning algorithms in a variety of tasks, and as a result, it is widely used. However, like other machine learning algorithms, deep learning, and convolutional neural networks (CNNs) in particular, perform worse when the data sets present label noise. Therefore, it is important to develop algorithms that help the training of deep networks and their generalization to noise-free test sets. In this paper, we propose a robust training strategy against label noise, called RAFNI, that can be used with any CNN. This algorithm filters and relabels instances of the training set based on the predictions and their probabilities made by the backbone neural network during the training process. That way, this algorithm improves the generalization ability of the CNN on its own. RAFNI consists of three mechanisms: two mechanisms that filter instances and one mechanism that relabels instances. In addition, it does not suppose that the noise rate is known nor does it need to be estimated. We evaluated our algorithm using different data sets of several sizes and characteristics. We also compared it with state-of-the-art models using the CIFAR10 and CIFAR100 benchmarks under different types and rates of label noise and found that RAFNI achieves better results in most cases.

Keywords: *Deep learning*, *label noise*, *convolutional neural network*, *robust learning*

12.1 INTRODUCTION

Over the last few years, deep learning and Convolutional Neural Networks (CNNs) in particular have become progressively popular as they have been used in a variety of applications, especially in computer vision, and outperform other models [KSH12;

GTL+19b; OTH18]. Generally speaking, the networks that have been used in these types of applications have become deeper and deeper over the years, due to their high performance.

A common problem when dealing with real-world data sets in the context of supervised classification is label noise. The term 'label noise' refers to when some instances in the data set have erroneous labels, thus misleading the training of machine learning algorithms [SKP+20]. This type of noise can be present in the data set because it was labelled automatically using text labels from the Internet, or because not enough experts were available to label an entire data set. In either case, the rate of label noise can vary and can increase to large values [XXY+15; LHZ+18]. As a result, label noise has been extensively studied when using classical machine learning algorithms [FV14]. The two most used and studied types of label noise are symmetric and asymmetric noise. In symmetric noise, also called uniform noise, the labels are corrupted randomly and equally in all classes, independently of their true class. In asymmetric noise, the corruptions are dependent on the true class of the instances, but not on the instances themselves. This implies that the corruptions in the labels can be made so that the instances in one specific class are labelled as another specific class. Subsequently, asymmetric noise is more realistic than symmetric noise. In [FV14], the authors made a taxonomy of label noise in classification, where symmetric noise is called Noisy Completely at Random (NCAR) and asymmetric noise is called Noisy at Random (NAR). There is another type of noise, called Noisy Not at Random (NNAR), which is the most realistic, where the corruptions are dependent on the true class of the instances and the instances themselves.

The use of increasingly deeper networks implies the need for larger data sets to adequately train them. This fact has caused researchers to investigate ways to overcome the lack of data. One possible solution is to create larger data sets by labelling them automatically instead of relying on experts, which is usually the only labelling solution for very large data sets [XXY+15; LWL+17; SKL19]. Another solution is to use techniques such as transfer learning and data augmentation. Transfer learning allows the network to start from a pre-trained state: instead of starting the training from scratch for every problem, the network is already pre-trained on another data set, usually larger and related to the new one in some way. As a consequence, transfer learning speeds up the training process. On the other hand, data augmentation artificially increases the size of the training data by introducing transformations of the original images, such as rotations, changes in lightning, cropping, flipping, etc.

But, if the original training set presents label noise, the use of data augmentation can aggravate it, thus becoming another source of label noise.

In the specific case of deep learning, label noise has been proven to harm generalization when training deep neural networks [ZBH+21], and thus there has been an increasing amount of studies trying to improve the behaviour of deep neural networks as much as possible in presence of label noise [PRK+17; SKL19; SKP+20; MWH+18; JZL+18]. Label noise appears mostly in real-world data sets, where the noise rate is not known. However, to test the performance of the proposed models, we need a controlled environment where the noise is artificially introduced in some noise-free data sets using different noise rates. Two of the most used data sets for this are CIFAR10 and CIFAR100 [WLM+18; SC18; JNC16; PRK+17]. Though it is necessary to test the proposals in large data sets like CIFAR or MNIST, it is also important to analyse them in other scenarios, like with small data sets. Small data sets are also common in real-world problems where it is not possible to collect more data. The majority of the current proposals lack this scenario.

To make the training of deep neural networks robust to all types of label noise, the analysis of incorrect classified instances for filtering and relabelling is the usual way to proceed. In this paper, we consider this hypothesis to deal with this problem, we consider our algorithm can identify and handle the noisy instances.

We propose an algorithm that, during the training process, relabels or filters the instances that it considers noisy using the predictions made by the backbone network. The backbone network is the deep neural network chosen to classify the data set (for instance, ResNet50), which will be trained using backpropagation as usual. Thus, we called the algorithm the Relabelling And Filtering Noisy Instances (RAFNI) algorithm. We have made the algorithm publicly available. As opposed to some of the previous proposals for this task, we do not suppose that the noise rate is known nor do we estimate it. Instead, the RAFNI algorithm uses the noisy training set and progressively cleans it during the training process by using the loss value of each instance and the probability of each instance belonging to each class. These values are given by the backbone network at each epoch of the training process. The algorithm is composed of two filtering mechanisms to remove noisy instances from the training set, and one relabelling mechanism that is used to change the class of some instances to their original (clean) class.

We evaluated our proposal with a variety of data sets, including small and large data sets, and under different types of label noise. We also use CIFAR10 and CIFAR100

as benchmarks to compare our proposal with other state-of-the-art models, since these data sets are two of the most used in other studies.

The rest of the paper is organized as follows. In Section 12.2, we give a background on works that propose strategies to help neural networks learn with label noise, with special mention to the ones we compare our algorithm to. In Section 12.3, we provide a detailed description of the RAFNI algorithm and its hyperparameters. Section 12.4 details the experimental framework, including the data sets, the types and levels of noise and the network configurations we used. The complete results obtained for all data sets and the comparison with the state-of-the-art models are shown in Section 12.5 and Section 12.6, respectively. We also compared our algorithm with an algorithm that supposes the noise rate is known and we show the results in Section 12.7. Then, in Section 12.8, we analyse the effectiveness of the RAFNI algorithm on two of the data sets used in this study. Finally, we give some final conclusions in Section 12.9.

12.2 BACKGROUND

In this section, we provide some background in the context of label noise and the types of noise we use in this study (Subsection 12.2.1). Then, we present an overview of the most popular approaches made in the context of deep learning to overcome the problem of label noise and provide a description of the proposals we have selected to compare with RAFNI (Subsection 12.2.2).

12.2.1 *Definition and types of label noise*

In the context of supervised classification, we have a set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n training instances and their corresponding labels $\mathbf{y} = (y_1, \ldots, y_n)$, where $y_i \in \{1, 2, \ldots, K\}$, $i = 1, \ldots, n$ and K is the total number of classes. Label noise presents when some instances in $\mathbf{X}$ have erroneous labels. That is, an instance $\mathbf{x}_i \in \mathbf{X}$ with a true label y_i actually appears in the training set with another label $\widetilde{y}_i$, $y_i \neq \widetilde{y}_i$, $i \in \{1, \ldots, n\}$. The percentage of instances that present label noise is called the noise rate or noise level.

Depending on whether the label noise appears dependent or independent of the class of the instances and the instances themselves, we can distinguish between the following types of noise:

- Symmetric noise (also called uniform noise or NCAR). The noise is independent of the original true class of the instances and the attributes of the instances. Thus, the labels of a percentage of the instances of the training set are randomly changed to another class following a uniform distribution, where all the classes have the same probability of being the noisy label. This implies that the percentage of noisy instances is the same in all classes. There are two options for this type of noise: the noisy label is chosen from the set of all of the classes (thus existing the possibility of not changing the label), or the noisy label is chosen from the set of all classes except the original one. We chose the second option.
- Asymmetric noise (also called NAR). The noise is dependent on the original true class of the instances and independent of the attributes of the instances. Therefore, the probability of each class to be the noisy label is different and depends on the original true class, but all the instances in the same class have the same probability of being noisy. This implies that the percentage of noisy instances in each class can be different.
- NNAR. The noise is dependent on the original true class of the instances and on the attributes of the instances. Hence, the probability of each class to be the noisy label can be different, depends on the original class, and the instances in each class have different probabilities to be noisy as it depends on the instances themselves.

As it happens in classical machine learning, the types of noise that we used can be treated (either by filtering or relabelling) without a prior estimation of the probability distribution.

12.2.2 *Label noise with deep learning*

During the last few years, there has been an increment in the number of proposals to help deep neural networks, and CNNs in particular, to learn in the presence of label noise in supervised classification. Most works fall into one or more of the following approaches:

- Proposals that modify the loss function in some way, either to make the loss function robust to label noise [AWA+19; GKS17; ZS18], or to correct its values, so the noisy labels do not negatively impact the learning [PRK+17; YW19; MWH+18; SKL19; AOA+19].

- Proposals that create a specific deep network architecture [XXY+15] or modify an existing one by adding a noise adaptation layer at the end of the desired architecture to model the noise [SBP+15; JNC16].
- Proposals that try to correct the noisy instances [PRK+17; SKL19].

Some proposals suppose that a subset of clean samples is available [XXY+15], and others assume that the noise rate is known [PRK+17; SKL19], which is not usual when dealing with real-world noisy datasets, though in [PRK+17] the authors propose a mechanism to approximate the noise rate. A more in-depth survey of all the work that has been done to learn deep neural networks in presence of label noise can be found in [SKP+20]. However, it is important to note that the majority of these proposals are highly focused on classifying the available benchmarks (such as CIFAR10/100 or TinyImagenet), and they use specific networks designed for CIFAR (ResNet32 or ResNet44) along with specific learning schedules. As a result, they are sometimes not generalizable to real-world problems.

We have selected a subset of five of these proposals to compare with our algorithm: one that uses a robust loss function, three that propose loss correction approaches, and one that proposes a hybrid approach between loss correction and sample selection. We choose them because they have official public implementations either on TensorFlow/Keras or PyTorch. In the following, we describe these five proposals.

1. Robust loss function approach that uses a generalization of the softmax layer and the categorical cross-entropy loss [AWA+19]. Here, the authors propose to make the loss function robust against label noise by modifying the loss function and the last softmax activation of the deep neural network with two temperatures, creating non-convex loss functions. These two temperatures can be tuned for each data set. This proposal has the advantage that using the code provided by the authors, it can easily be used with any combination of a deep network, data set and optimization technique, including transfer learning.

2. Loss correction approaches [PRK+17]. The authors propose two approaches to correct the loss values of the noisy instances, for which it is necessary to know the noise matrix of the data set, called backward correction and forward correction. They provide a mechanism to estimate the noise matrix, and when used, the approaches are called estimated backward correction and estimated forward correction. The first one uses the noise matrix to correct the loss values,

so they are more similar to the loss values of the clean instances. The second explicitly uses the noise matrix to correct the predictions of the model.

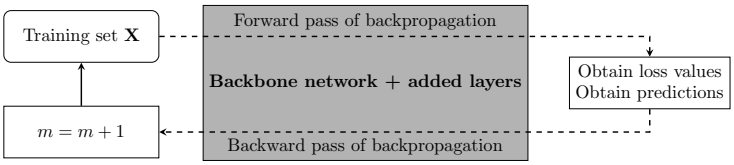
3. Loss correction approach using the dimensionality of the training set [MWH+18]. The authors explain that, when dealing with noisy labels, the learning can be separated into two phases. In the first phase, which occurs in the first epochs of the training, the network learns the underlying distribution of the data. Then, in the second phase, the network learns to overfit the noisy instances. They use a measure called Local Intrinsic Dimensionality (LID) to detect the moment the training enters the second phase and use the LID to modify the loss function to reduce the effect of the noisy instances.

4. Loss correction approach [AOA+19], which is based on the static hard bootstrapping loss proposed in [RLA+14] combined with a data augmentation technique called mixup proposed in [ZCD+17]. They use a beta mixture model to fit the loss values of the instances so they can distinguish between clean and noisy instances and use the loss correction approach on the noisy ones.

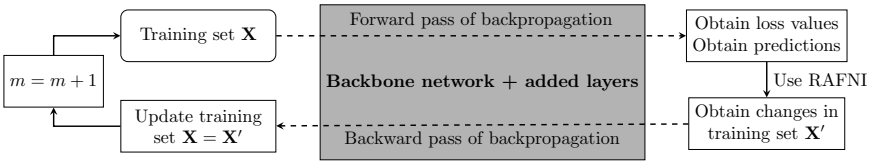
5. Loss and label correction approach [SKL19]. The authors propose a hybrid approach between sample selection and loss correction that tries to relabel noisy instances when possible and not use them when not. For the noisy instances, they rely on the network: if it returns the same label with a high probability in the first epochs of the training, it is possible to correct that instance and the algorithm changes its label to the one the network predicts. In contrast, if the network changes the prediction of an instance inconsistently, they stop using that instance. They assume that the noise rate in the data set is known, and they do not provide a way to estimate it. This approach can be used iteratively so that the training set is iteratively cleaned in several training processes.

12.3 RAFNI: RELABELLING AND FILTERING INSTANCES BASED ON THE PREDICTIONS OF THE BACKBONE NETWORK

In this section, we describe our proposal. First, in Subsection 12.3.1, we give an overall description of the algorithm and explain its basics. Then, in Subsection 12.3.2, we present a formal definition of the algorithm. Finally, in Subsection 12.3.3 we give a guide on how to tune the hyperparameters of the algorithm.



(a) The standard training process of the backbone network for an epoch m



(b) The training process of the backbone network using RAFNI for an epoch m

Figure 12.1: Difference between training the backbone network (a) without and (b) with the RAFNI algorithm

12.3.1 *Base concepts*

We propose the RAFNI algorithm, which filters and relabels instances based on the predictions and their probabilities made by the backbone neural network during the training process. In Figure 12.1, we show the difference between training the backbone network with and without the RAFNI algorithm and the moment it is applied. The backbone network used is independent of the algorithm, and it can change or be modified, for example, including transfer learning.

Generally speaking, we propose two mechanisms to filter an instance and one mechanism to relabel an instance, with some restrictions. These mechanisms are the following:

- First filtering mechanism. This mechanism only uses the loss value of the instances. The foundation is that the noisy instances tend to have higher loss values than the rest of them. As a result, this mechanism filters out instances that have a loss value above a certain threshold. This threshold is dynamic and will change during training.

- Second filtering mechanism. This mechanism depends on how many times an instance has been relabelled. Here we suppose that if the algorithm relabels an instance too many times is because the backbone network is unsure about its class and it is better to remove that instance. Thus, this mechanism filters an instance if it has been relabelled more than a certain number of times. In addition, we establish a period of a certain number of epochs after an instance has been relabelled during which the algorithm cannot filter nor relabel it again.

- Relabelling mechanism. This mechanism takes into account the probability predictions of the backbone network. We suppose that if the backbone network predicts another class with a high probability as the training progresses, it is probable that the instance is noisy and its class is indeed the one predicted by the backbone network. As a consequence, the relabelling mechanism changes the class of an instance if the backbone network predicts another class with a probability that is above a certain threshold. This threshold is also dynamic and will change during training.

These mechanisms have restrictions related to the moment they are applied. Since we are using the backbone network to relabel and filter instances, we need to wait until the network is sufficiently trained for the predictions to be reliable. This can be measured using the loss values of the instances. Intuitively, we want to start the algorithm (and thus the three mechanisms) when the backbone network has learned to classify the clean instances but it has not learned to overfit the noisy ones yet. Here, similarly to [AOA+19], we approximated the loss values of the instances in each epoch of the training process by a mixture model with two components, but in our case, we use a Gaussian mixture model. To do that, we used the expectation minimization algorithm and used the two components to detect the moment where the RAFNI algorithm needs to start.

A mixture model is a model that can represent different subpopulations inside a population. These subpopulations or components follow a distribution that in a Gaussian mixture model is supposed to be a Gaussian distribution. That way, if we have a Gaussian mixture model with two components, we are approximating two subpopulations, each one with a Gaussian distribution, so we obtain two means and two variances. In our case, we have two components, one for the clean instances, with mean μ_{clean} and standard deviation σ_{clean}, and one for the noisy instances, with mean μ_{noisy} and standard deviation σ_{noisy}.

We detect the moment we need to start the RAFNI algorithm by calculating the overlap between the two Gaussians obtained by the mixture model over the loss values of the instances in all training epochs. At first, the two Gaussians will start separating from one another, while the network learns to classify the clean and easy examples. Then, at some point, they will start to get closer, as the network starts to overfit the noisy examples. Therefore, we start the algorithm when the overlap between the Gaussians is below a fixed value or when this overlap starts to increase. We have tested different values for this hyperparameter with different data sets and levels of noise and we found that 0.15 is a good value that can remain fixed across all data sets and noise rates.

12.3.2 *Formal definition*

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be the set of n training instances and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ their corresponding labels, where $y_i \in \{1, 2, \ldots, K\}$, $i = 1, \ldots, n$, and K is the total number of classes. Let m be an epoch of the training process, $m = 1, \ldots, M$, where M is the total number of epochs, and $\mathbf{l}_m = (l_{m1}, l_{m2}, \ldots, l_{mn})$ the losses of the training instances in epoch m, $m = 1, \ldots, M$. Finally, let $\mathbf{p}_{mi} = (p_{mi1}, p_{mi2}, \ldots, p_{miK})$ be the probabilities predicted by the backbone neural network for each instance $\mathbf{x}_i \in \mathbf{X}$ in epoch m, and $\widehat{y}_{mi} \in \{1, 2, \ldots, K\}$ the prediction of the backbone network for the instance $\mathbf{x}_i$ in epoch m, where $m = 1, \ldots, M$, $i = 1, \ldots, n$. Then, we define the following:

1. A threshold, named `loss_threshold`, so that if $l_{mi} >$ `loss_threshold`, then $\mathbf{x_i}$ is removed from the training set for the following epochs $m + 1, \ldots, M$, where $i = 1, \ldots, n$.

2. A number, `record_length`, denoting the length of the `record` of each instance, so that the algorithm saves the last `record_length` predictions made by the neural network in the last `record_length` epochs of the training. Then, if the predictions of an instance $\mathbf{x}_i$ change `record_length`-1 times in the last `record_length` epochs, the instance $\mathbf{x}_i$ is removed from the training set for the following epochs $m + 1, \ldots, M$, where $i = 1, \ldots, n$.

3. A threshold, `prob_threshold`, so that if $\max_k(\mathbf{p}_{mi}) >$ `prob_threshold` and $y_i \neq \widehat{y}_{mi}$, then $y_i = \widehat{y}_{mi}$ in the following epochs $m + 1, \ldots, M$, where $i = 1, \ldots, n$. If this happens, the algorithm clears the `record` of the instance $\mathbf{x}_i$.

4. A number, `not_change_epochs`, so that if the label of an instance has been changed, the algorithm cannot change it again nor remove that instance from the training set until `not_change_epochs` epochs have passed.

In Figure 12.2, we show the flowchart of the RAFNI algorithm, detailing how and when each mechanism is applied to each instance $\mathbf{x}_i$ during a specific epoch m of the training process.

The numbers `record_length` and `not_change_epochs` are hyperparameters of the algorithm that can be setted by the user. The two thresholds, `loss_threshold` and `prob_threshold`, are parameters that dynamically change every epoch m using the losses of the instances and their probabilities in the previous epoch, l_{m-1}. Specifically, the `loss_threshold` is calculated for every epoch as the quantile of order x_1 (x_1 is an hyperparameter of the algorithm called `quantile_loss`, that can be setted by the user) of the losses in the previous epoch l_{m-1}. Similarly, the `prob_threshold` is calculated for every epoch as the quantile of order x_2 (also a hyperparameter, called `quantile_prob`) of the probabilities returned by the backbone network for the misclassified instances. That way, the `loss_threshold` usually descends as the epoch increases and the training instances are being filtered and their classes relabelled. Due to that, we need to stop updating the `loss_threshold` parameter at some point to not filter too many instances. Similarly, we also need to stop updating the `prob_threshold` parameter so it does not change the class of too many instances. To do this, we use again the two Gaussians obtained by the Gaussian mixture model and stop the update of both thresholds when the means of the Gaussians are sufficiently close. We tested different values and we obtained that 0.3 is a good value that works for different data sets and levels of noise. That way, we stop updating the thresholds if $\mu_{\text{noisy}} - \mu_{\text{clean}} < 0.3$.

This algorithm can be used with any CNN as a backbone network. The code of the algorithm is available at https://github.com/ari-dasci/S-RAFNI.

12.3.3 *A guide to the hyperparameters of RAFNI*

RAFNI has a list of hyperparameters that can be fine-tuned by the user. Here we specify which hyperparameters are most important to be tuned if a validation set is available, which ones are less important and which ones do not need to be tuned. We also give a guide on how to tune them.

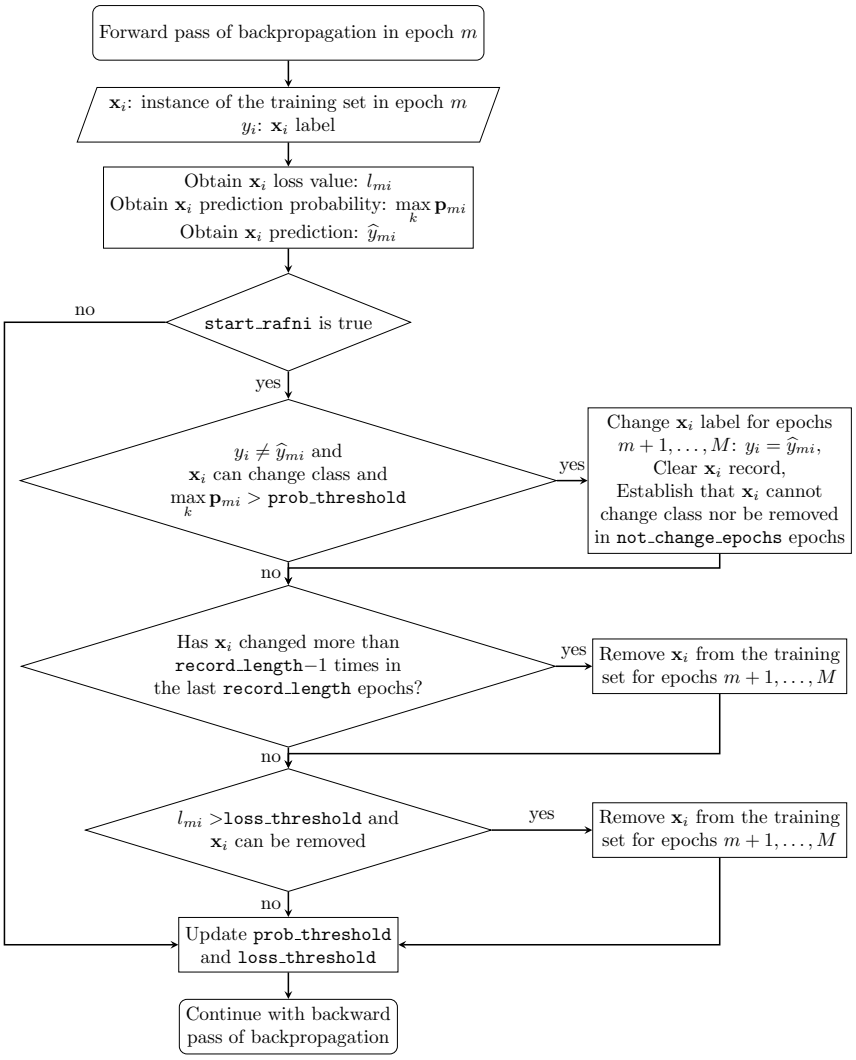


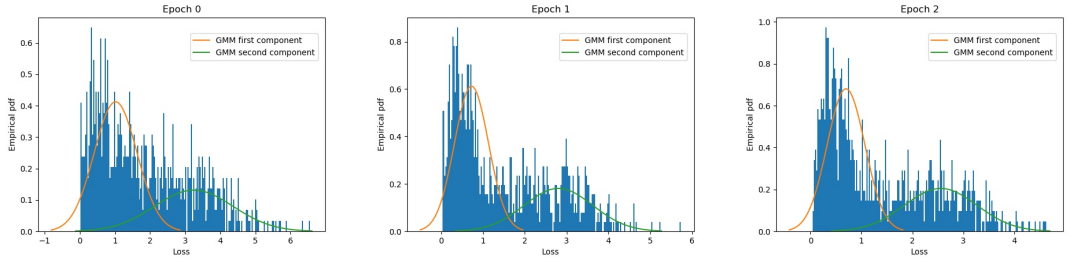
Figure 12.2: Flowchart of the RAFNI algorithm

The complete list of hyperparameters of RAFNI is the following: the overlapping threshold between the noisy Gaussian and the clean Gaussian we use to start the algorithm, the difference between the means of the two Gaussians we use to stop

the update of the `loss_threshold` and the `prob_threshold`, the `quantile_loss`, the `quantile_prob`, the `record_length` and the `not_change_epochs`. The `loss_threshold` and the `prob_threshold` are not really hyperparemeters as they cannot be tuned, they change dinamically in each epoch based on the `quantile_loss` and `quantile_prob` hyperparameters, respectively.

We also have the hyperparameters inherent to training a deep neural network: the total number of epochs of the training, the batch size and whether to use fine-tune or not in the backbone CNN: if we do not use fine-tuning, the layers of the backbone neural network are not retrained and only the new added layers are trained, and if we use fine-tuning, all the layers are trained. Whether to use fine-tuning or not depends on the backbone network used (how deep it is) and if the data set we want to classify has enough images to retrain the whole network or not. The number of epochs of the training depends on if we are fine-tuning the backbone network and the size of the data set. The batch size depends on the size of the data set, it will increase as the size of the data set increases, usually.

If we focus on the specific hyperparameters of RAFNI, there are two of them that we recommend not changing: the overlapping threshold between the Gaussians we use to start the algorithm and the difference between the means of the Gaussians that we use to stop the updates of `loss_threshold` and `prob_threshold`. We tested different values for these parameters across all the data sets and levels of noise we used and we found that 0.15 is a good value for the overlapping threshold and 0.3 a good value for the difference between the means of the Gaussians. To show why we chose these values we can see Figures 12.3, 12.4 and 12.5. In Figure 12.3 we show the two components (the two Gaussians) obtained by the Gaussian Mixture Model (GMM) over the losses of the instances in the first epochs of the training of EILAT with 40% of symmetric noise. At first, as the learning progresses, the network starts to differentiate between the clean and the noisy instances, and thus the two components start to separate from themselves. Then, the network starts to overfit the noisy instances and the two components start to come together again. To stop this overfitting, we start the RAFNI algorithm when the overlap between the two components is less than 0.15 or when the overlap in an epoch is greater than in the previous epoch. In Figure 12.4 we can see how this overlap changes through the epochs of the training and in which specific epoch we are starting the RAFNI algorithm. Finally, to stop the updating of the `loss_threshold` and the `prob_threshold` we use the distance between the means of the two components: if they are close, it means that there are not enough noisy instances, so the two components are very close. In Figure 12.5 we



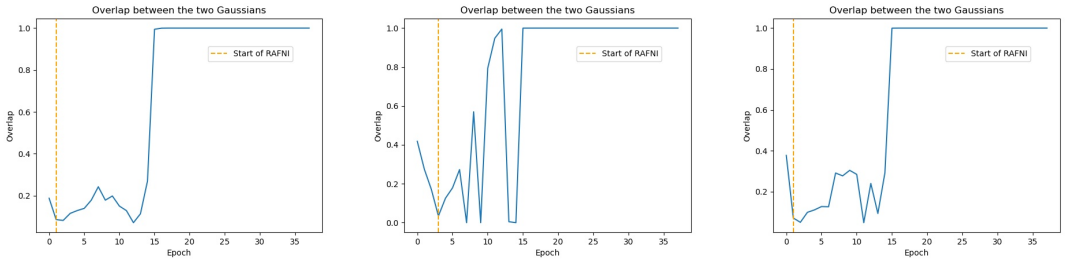
(a) Epoch 0 (b) Epoch 1 (c) Epoch 2

Figure 12.3: The two components obtained by the Gaussian Mixture Model (GMM) over the loss values of the instances in the first three epochs of the training using the EILAT data set at 40% of noise.

can see the evolution of the difference between the means of the two Gaussians and the specific epoch where we stop the updating of the thresholds.

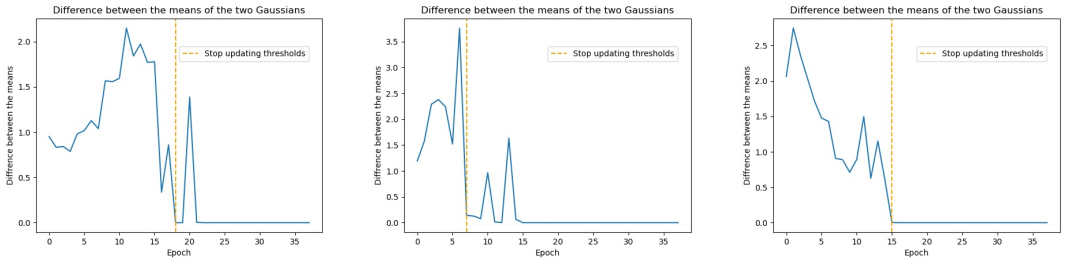
Regarding the rest of the hyperparameters of RAFNI, we recommend to tune them if possible, though we found that tunning the `quantile_loss` and `quantile_prob` is more important than tunning the `record_length` and `not_change_epochs` hyperparameters. The `quantile_loss` depends on how hard is the data set to classify: the more difficult it is, the less we can rely on the predictions of the backbone network and it is more convenient to use higher values so that the algorithm is more conservative, that is, so it does not remove nor change the class of too many instances. Something similar happens with the values of the hyperparameters `quantile_prob`, `record_length` and `not_change_epochs`, though in these cases is also related to the size of the data set: the harder it is to classify and fewer images have, higher values we should give to these hyperparameters. In the case of `record_length` and `not_change_epochs`, we should also take into account the total number of epochs of the training: if this number is small, these two hyperparameters should also be small and they should not exceed the total number of epochs in any case).

The ranges in which each hyperparameter can take values are the following. For the `quantile_loss` and the `quantile_prob` hyperparameters, given they are quantiles, their maximum value is 1 and their minimum is 0. However, we found that they perform best if they vary in the range $[0.6, 0.99]$. The minimum value for `record_length` is 2, so it can track at least one change in the class of the instances, and its maximum is the total number of epochs in the training; the higher this value



(a) EILAT at 40% noise (b) StructureRSMAS at 10% noise (c) RSMAS at 30% noise

Figure 12.4: Evolution of the overlap between the two components of the GMM through the epochs of the training of different data sets and noise rates.



(a) EILAT at 70% noise (b) StructureRSMAS at 10% noise (c) RSMAS at 30% noise

Figure 12.5: Evolution of the difference between the means of the two components of the GMM through the epochs of the training of different data sets and noise rates.

is, the fewer instances the algorithm will remove because their class has changed. Finally, the minimum value for `not_change_epochs` is 1 and the maximum is the total number of epochs in the training.

12.4 EXPERIMENTAL FRAMEWORK

In this section, we describe the experimental framework we used to carry out the experiments. In Subsection 12.4.1, we describe the data sets we used. In Subsection 12.4.2, we detail the types of noise we used in each data set along with the noise levels we used in each one of them. Finally, in Subsection 12.4.3, we provide the specific configuration, backbone neural network and software we used for all the experiments.

12.4.1 *Data sets*

We describe the data sets we used to analyse RAFNI under different types and levels of label noise. We used six data sets, each one with a different number of classes, images per class and a total number of images: RSMAS, StructureRSMAS, EILAT, COVIDGR1.0-SN, CIFAR10 and CIFAR100. There is a summary of the statistics of these data sets in Table 12.1.

RSMAS, StructureRSMAS and EILAT are small coral data sets. RSMAS and EILAT [Shi; GTL+19b] are texture data sets, containing coral patches, meaning that they are close-up patches extracted from larger images, and StructureRSMAS [GTL+19a] is a structure data set, containing images of entire corals. The patches in EILAT have a size of 64×64 and come from images taken under similar underwater conditions, and the ones in RSMAS have a size of 256×256 and come from images taken under different conditions. StructureRSMAS is a data set collected from the Internet and therefore contains images of different sizes taken under different conditions.

COVIDGR1.0-SN is a modification of COVIDGR1.0 [TGM+20]. COVIDGR1.0 contains chest x-rays of patients divided into two classes: positive for COVID-19, and negative for COVID-19, using the RT-PCR as ground truth. All the images in the data set were taken using the same protocol and similar x-ray machines. The authors made available the data set along with a list containing the degree of severity of the positive x-rays: Normal-PCR+, Mild, Moderate and Severe. The x-rays with Normal-PCR+ severity are x-rays from patients that tested positive in the RT-PCR test but where

Table 12.1: A summary of the data sets used in this study.

Data set	**# classes**	**# images**	**# images per class**
RSMAS	14	766	1: 109, 2: 77, 3: 57, 4: 63, 5: 24, 6: 60, 7: 22, 8: 79, 9: 54, 10: 28, 11: 32, 12: 88, 13: 37, 14: 36
StructureRSMAS	14	409	1: 44, 2: 41, 3: 34, 4: 20, 5: 16, 6: 18, 7: 33, 8: 38, 9: 30, 10: 21, 11: 32, 12: 23, 13: 36, 14: 23
EILAT	8	1123	1: 87, 2: 78, 3: 29, 4: 160, 5: 200, 6: 216, 7: 296, 8: 11.
COVIDGR1.0-SN	2	700	350
CIFAR10	10	60000	6000
CIFAR100	100	60000	600

Table 12.2: Types and levels of label noise used for each data set

Data set	Type of label noise	Levels of noise
RSMAS	Symmetric	0%, 20%, 30%, 40%, 50%, 60% and 70%
StructureRSMAS		
EILAT		
COVIDGR1.0-SN	NNAR	0%, 20%, 30%, 40% and 50%
CIFAR10	Asymmetric	0%, 20%, 30% and 40%
CIFAR10	Symmetric	0%, 20%, 40% and 60%
CIFAR100		

expert radiologists could not find signs of the disease in the x-ray. The modification that we use, COVIDGR1.0-SN, is the same data set as COVIDGR1.0 but we removed the 76 positive images with Normal-PCR+ severity. To maintain the two classes balanced, as happens in the original data set, we also removed 76 randomly chosen negative images.

Finally, CIFAR10 and CIFAR100 [KH+09] are the 60k tiny images of size 32×32 images proposed by Alex Krizhevsky. Concerning the other data sets used in this study, CIFAR10 and CIFAR100 are much larger in size. Both of them have a predefined test hold-out of 10.000 images, meaning they both have a training set of 50.000 images. Both datasets contain classes of common objects, such as 'Airplane' and 'Ship' in CIFAR10 or 'Bed' and 'Lion' in CIFAR100.

12.4.2 *Types and levels of label noise*

We state the types of noise we used for each data set and which rates we used for each one of them. In Table 12.2 we show a brief description. In summary, we used symmetric noise, asymmetric noise and NNAR noise. Symmetric noise is the most used type of noise and since it is not necessary to have external information to use it, we use this type of noise in all data sets except for COVIDGR1.0-SN. But, to also use more realistic and challenging types of noise, we used asymmetric and NNAR noise when possible. This is the case for CIFAR10 and COVIDGR1.0-SN.

For CIFAR10, we used the asymmetric noise introduced in [PRK+17], which has been a standard when evaluating deep learning in the presence of asymmetric label

noise. This noise is introduced between classes that are alike, simulating real label noise that could have occurred naturally. In particular, we introduced asymmetric noise between the following classes: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG, as defined in [PRK+17]. Note that since we are introducing an x% of noise in five of the ten classes, we are introducing an $\frac{x}{2}$% of noise in the total dataset.

For COVIDGR1.0-SN, we have the additional information on the severity degree in the images of the positive class, so we used them to introduce NNAR noise, where we change the labels of a percentage x of the instances of the data set subject to some condition over the instances. COVIDGR1.0-SN has two classes: P (COVID-19 positive) and N (COVID-19 negative), and the instances from P have associated a severity (Mild, Moderate and Severe). In this scenario, is more realistic that a positive image with mild severity has been misclassified as negative than a positive image with moderate or severe severity. Equivalently, is more realistic that a positive image with moderate severity has been misclassified as negative than a positive image with severe severity. As a consequence, we define the probability of the instances in the groups N (to change class to P), Mild (to change class to N), Moderate (to change class to N) and Severe (to change class to N) as it follows: 0.5 for N, 0.3 for Mild, 0.2 for Moderate and 0 for Severe. That way, we are changing the same amount of instances from P to N and vice versa, but when we change the class from P to N, it is more probable to change a mild positive image than a moderate positive image. In addition, we are making sure that no positive image with severe severity has changed from class P to N.

12.4.3 *Network and experimental configuration*

Here, we provide the specific configuration we used in the experiments we carried out.

As the backbone neural network we used ResNet50 [HZR+16], though the implementation of RAFNI is independent of the backbone network and can be changed easily. We used ResNet50 pre-trained using ImageNet, removing the last layer of the network and adding two fully connected layers, the first one with 512 neurons and ReLU activation and the second one with as many neurons as classes the data set had and softmax activation. Once we removed the last layer of ResNet50, its output had 2048 neurons. We chose 512 neurons for the first fully connected layer

Table 12.3: Fixed hyperparameters we used in each data set.

Data set	Optimizer	Batch size	Total number of epochs	Fine-tune
RSMAS	SGD	16	40	No
StructureRSMAS	SGD	16	40	No
EILAT	SGD	16	40	No
COVIDGR1.0-SN	SGD	16	40	No
CIFAR10	SGD	128	10	Yes
CIFAR100	SGD	128	15	Yes

we added as an intermediate number between 2048 and the number of classes in the data sets. The fixed hyperparameters we used in each data set can be seen in Table 12.3. We used the Stochastic Gradient Descent (SGD) with a learning rate of 1×10^{-3}, a decay of 1×10^{-6} and a Nesterov momentum of 0.9. We did not optimize these hyperparameters.

For the experimentation, we used TensorFlow 2.4 and an Nvidia Tesla V100. The values we gave to each hyperparameter can be seen in Table 12.4. Using those values, we performed an exhaustive grid search to find the best configuration in each case. Since the CIFAR data sets and the rest of the data sets we used had different sizes, the experimental framework we used for them was different. For the smaller data sets RSMAS, EILAT, StructureRSMAS and COVIDGR1.0-SN, we used five-fold cross-validation for the experiments in the grid search, while for CIFAR10 and CIFAR100 we used a 20% hold-out using only the original train set. Then, to ensure a more stable final result, we did the following. For the smaller data sets, we repeated the five-fold cross-validation with the best hyperparameter configuration five times (noted 5x5fcv) and we report the mean and standard deviation of the 5x5fcv. This scheme of using mean and standard deviation is one of the most used in the literature. For the CIFAR data sets, we used the best hyperparameter configuration (found using only the training set) in the predefined test hold-out, we repeated that experiment five times and we report the mean and standard deviation.

We used the accuracy measure, widely used for supervised classification. The accuracy is defined as the number of instances well classified in the test set divided by the total number of instances in the test set.

Table 12.4: Values we used for each hyperparameter and data set for the grid search.

Data set	Hyperparameter	Values
RSMAS, EILAT, StructureRSMAS and COVIDGR1.0-SN	`quantile_loss`	{0.9, 0.92, 0.94, 0.96, 0.98, 0.99}
	`quantile_prob`	{0.9, 0.93, 0.95, 0.97, 0.99}
	`record_length`	{5, 8}
	`not_change_epochs`	{4, 7}
CIFAR10	`quantile_loss`	{0.95, 0.97, 0.99}
	`quantile_prob`	{0.75, 0.8, 0.85, 0.9}
	`record_length`	{2, 4}
	`not_change_epochs`	{1, 2}
CIFAR100	`quantile_loss`	{0.95, 0.97, 0.99}
	`quantile_prob`	{0.75, 0.78, 0.82, 0.85}
	`record_length`	{2, 4}
	`not_change_epochs`	{1, 2}

To make the comparison with the baseline model (that is, the backbone network alone, without filtering or relabelling instances), we used the same scheme as we used with RAFNI: we repeated five times the five-fold cross-validation (or the hold-out for the CIFAR data sets) and we report mean and standard deviation.

The best values for the hyperparameters in each case can be found in Table 12.5.

12.5 COMPARISON WITH THE BASELINE MODEL

In this section, we present the results we obtained for each data set using our proposal, and we compare it with the backbone network alone as the baseline.

12.5.1 *RSMAS, EILAT, StructureRSMAS and COVIDGR1.0-SN*

We present the results obtained with the smaller data sets: RSMAS, EILAT, StructureRSMAS and COVIDGR1.0-SN.

Table 12.5: Best hyperparameter values for all data sets using ResNet50.

Data set	Noise rate	quantile_loss	quantile_prob	record_length	not_change_epochs
RSMAS	0%	0.99	0.95	5	7
	10%	0.96	0.93	5	7
	20%	0.92	0.95	8	4
	30%	0.92	0.9	5	7
	40%	0.9	0.95	5	7
	50%	0.9	0.93	5	7
	60%	0.9	0.9	8	4
	70%	0.9	0.93	5	4
StructureRSMAS	0%	0.99	0.9	8	4
	10%	0.96	0.95	8	7
	20%	0.94	0.95	5	4
	30%	0.92	0.95	5	4
	40%	0.92	0.95	5	4
	50%	0.92	0.95	8	4
	60%	0.9	0.95	5	4
	70%	0.9	0.9	5	7
EILAT	0%	0.99	0.99	8	4
	10%	0.98	0.97	5	7
	20%	0.94	0.9	8	7
	30%	0.94	0.95	8	4
	40%	0.92	0.97	8	7
	50%	0.9	0.93	8	4
	60%	0.9	0.95	5	4
	70%	0.9	0.9	5	4
COVIDGR1.0-SN	0%	0.96	0.99	8	7
	10%	0.99	0.9	5	7
	20%	0.94	0.97	5	4
	30%	0.96	0.97	5	4
	40%	0.92	0.93	5	7
	50%	0.9	0.95	5	7
CIFAR10, symmetric noise	0%	0.99	0.75	4	1
	20%	0.95	0.75	2	1
	40%	0.97	0.75	2	2
	60%	0.99	0.8	2	2
CIFAR10, asymmetric noise	0%	0.99	0.75	4	1
	20%	0.95	0.8	2	2
	30%	0.95	0.85	2	2
	40%	0.95	0.85	2	1
CIFAR100, symmetric noise	0%	0.95	0.85	4	2
	20%	0.95	0.78	4	1
	40%	0.99	0.75	2	1
	60%	0.99	0.75	2	1

Table 12.6: 5x5fcv mean ± std accuracy obtained for the data sets RSMAS, EILAT and StructureRSMAS using RAFNI with ResNet50 as backbone network and the backbone network alone, ResNet50, as baseline. The best results in each case are stressed in bold.

Data set	Algorithm	0%	10%	20%	30%	40%	50%	60%	70%
RSMAS	Baseline	**97.78 ± 0.94**	93.50 ± 1.60	88.04 ± 3.42	82.59 ± 2.85	74.10 ± 3.73	63.91 ± 3.72	52.54 ± 4.15	38.83 ± 5.64
	RAFNI	97.70 ± 1.39	**96.76 ± 1.49**	**95.51 ± 2.16**	**92.09 ± 1.68**	**88.95 ± 2.48**	**79.13 ± 3.89**	**67.04 ± 4.42**	**53.73 ± 4.68**
EILAT	Baseline	**97.53 ± 1.51**	94.65 ± 2.09	89.67 ± 1.81	84.10 ± 2.40	75.96 ± 4.44	65.44 ± 4.24	53.30 ± 4.11	37.46 ± 5.18
	RAFNI	97.41 ± 1.64	**96.37 ± 1.57**	**95.80 ± 1.94**	**95.34 ± 2.04**	**93.60 ± 2.10**	**90.76 ± 2.77**	**86.72 ± 3.34**	**76.67 ± 5.57**
StructureRSMAS	Baseline	81.73 ± 4.42	80.23 ± 4.06	74.37 ± 5.00	70.87 ± 5.39	62.59 ± 5.05	51.60 ± 7.01	40.90 ± 6.16	32.98 ± 4.06
	RAFNI	**82.05 ± 3.78**	**81.17 ± 3.63**	**77.40 ± 4.99**	**74.09 ± 5.35**	**68.43 ± 5.24**	**54.92 ± 6.22**	**46.57 ± 8.93**	**35.90 ± 6.04**

Table 12.7: 5x5fcv mean ± std accuracy obtained for the data set COVIDGR1.0-SN using RAFNI with ResNet50 as backbone network and the backbone network alone, ResNet50, as baseline. The best results in each case are stressed in bold.

Noise	Baseline	RAFNI
0%	77.06 ± 3.47	**78.20 ± 2.80**
10%	73.46 ± 2.38	**76.31 ± 2.44**
20%	72.71 ± 4.74	**75.91 ± 2.92**
30%	68.14 ± 4.00	**75.06 ± 3.94**
40%	62.49 ± 2.87	**72.77 ± 4.68**
50%	55.34 ± 4.33	**64.46 ± 5.68**

In Table 12.6, we can observe the results with symmetric noise for the data sets RSMAS, EILAT and StructureRSMAS using RAFNI with ResNet50 as the backbone network, and the comparison with the backbone network alone as the baseline. The results are similar in all data sets as the noise increases, especially for RSMAS and EILAT. At 0% of noise, the difference between the use of the RAFNI algorithm and the baseline is minimal. Then, as noise increases, this difference starts to increase. At 10% of noise, RAFNI obtains 3.26% more than the baseline for RSMAS and 1.72% for EILAT. At 40% of noise, this difference is 14.85% for RSMAS and 17.64% for EILAT. At 70% of noise, the gain of using RAFNI is 14.9% for RSMAS and 39.21% for EILAT. For StructureRSMAS, these differences are lower: for example, at 40%, the gain of using RAFNI is 5.84%. However, RAFNI is still consistently better than the baseline at all levels of noise.

Table 12.8: Mean ± std accuracy obtained using CIFAR10 and CIFAR100 with symmetric noise and using the baseline network (ResNet50) and RAFNI with that network as the backbone network. The best results in each case are stressed in bold.

Data set	CIFAR10				CIFAR100			
Noise	0%	20%	40%	60%	0%	20%	40%	60%
Baseline	95.31 ± 0.03	85.36 ± 0.41	67.94 ± 0.50	44.87 ± 0.34	**81.18 ± 0.26**	70.57 ± 0.25	55.88 ± 0.45	37.64 ± 0.47
RAFNI	**95.48 ± 0.09**	**92.86 ± 0.32**	**89.84 ± 0.66**	**79.48 ± 1.23**	80.68 ± 0.14	**77.93 ± 0.11**	**73.01 ± 0.16**	**67.08 ± 0.45**

The results obtained for COVIDGR1.0-SN with pseudo-symmetric noise are shown in Table 12.7. Here we only used levels of noise up until 50% because this data set has only two classes. This data set has the advantage that it is a real-world data set, and it is more difficult to train (at 0% noise level) than the other data sets: ResNet50 obtains an accuracy of 77.06% at 0% noise. In addition, the noise we introduced in this data set is more realistic, so we can see how well the RAFNI algorithm behaves in a more real-life scenario. We can see that, at all noise levels, including 0%, the results are better using RAFNI than using the baseline, with gains that generally increase as the noise level raises. At 10% noise, RAFNI obtains 2.85% more than the baseline. This gain is 6.92% at 30% noise and 9.12% at 50% noise.

12.5.2 *CIFAR*

We show the results we have obtained using CIFAR10 and CIFAR100 with symmetric noise and CIFAR10 with asymmetric noise.

In Table 12.8 we can see the results for CIFAR10 and CIFAR100 using symmetric noise. RAFNI achieves better results in both data sets at all levels of noise except for CIFAR100 at 0% of noise, where the baseline is slightly better. Similarly to what happened with the small datasets, the accuracy gain of using RAFNI increases as the noise level increases. For CIFAR10 we have a gain of 7.5% at 20% of noise and a gain of 34.61% at 60% of noise. For CIFAR100 these gains are 7.36% and 29.44% at 20% and 60% of noise, respectively.

In Table 12.9 we show the results for CIFAR10 using asymmetric noise. In this scenario, RAFNI also achieves better results than the baseline at all levels of noise. The accuracy gain of using RAFNI increases when the noise increases, being 4.69% at 20% of noise, 8.69% at 40% of noise and 10.41% at 60% of noise.

Table 12.9: Mean ± std accuracy obtained using CIFAR10 with asymmetric noise and using the baseline network (ResNet50) and RAFNI with that network as the backbone network. The best results in each case are stressed in bold.

Noise	0%	20%	30%	40%
Baseline	95.31 ± 0.03	89.27 ± 0.43	84.27 ± 0.30	78.10 ± 0.52
RAFNI	**95.48 ± 0.09**	**93.96 ± 0.15**	**92.96 ± 0.13**	**88.51 ± 0.54**

12.6 COMPARISON WITH STATE-OF-THE-ART MODELS

In this section, we compare our proposal, RAFNI, with some state-of-the-art models that do not used external information (like the noise rate): the loss correction approaches proposed in [PRK+17], the robust function proposed in [AWA+19], the proposal in [MWH+18], and the one in [AOA+19], which we described in Section 12.2.

The two proposed methods in [PRK+17] suppose that the noise rate is known, but the authors incorporated a mechanism to estimate it in the usual case that it is not known, so we used both of their approaches using this estimation (called estimated forward and estimated backwards).

To make the comparison we used CIFAR10, with symmetric and asymmetric noise, and CIFAR100 with symmetric noise, which are the benchmarks that most of the papers used in the literature.

To make a fair comparison, we used the same experimental frameworks as the other papers whenever possible, that is, we changed our framework to use the same backbone neural network, the same number of training epochs, optimizer, the same learning rate scheduler, data augmentation, etc., as the model we are comparing our algorithm to. In each case, we used the best hyperparameters reported in each paper for each scenario, except for the number of epochs. Due to time restrictions, if the original number of epochs used by the authors exceeds 120 for CIFAR10 and 150 for CIFAR100, we changed them to use 120 epochs for CIFAR10 and 150 epochs for CIFAR100 (accordingly, we also train RAFNI for the same number of epochs in each case). The authors in [AWA+19] only used CIFAR100 under symmetric noise, so we only had the best values for the two temperatures for this case. For the other two scenarios (CIFAR10 with symmetric noise and with asymmetric noise), we evaluated different values in the range the authors gave for each hyperparameter and selected the best ones for each scenario and level of noise using the same validation set we

Table 12.10: Comparison between RAFNI and the two methods from Patrini et al [PRK+17], using pre-activation ResNet32 for CIFAR10 and pre-activation ResNet44 for CIFAR100 in the three approaches. The best results are stressed in bold.

	CIFAR10						CIFAR100		
	Symmetric noise			Asymmetric noise			Symmetric noise		
	20%	40%	60%	20%	30%	40%	20%	40%	60%
Est. Forward	**88.46 ± 0.22**	**84.54 ± 0.40**	79.32 ± 0.23	**89.89 ± 0.15**	**89.32 ± 0.28**	87.09 ± 1.55	**62.11 ± 2.57**	48.72 ± 0.84	33.41 ± 0.87
Est. Backwards	84.40 ± 0.22	79.12 ± 0.45	64.00 ± 1.67	86.33 ± 0.55	81.71 ± 3.06	72.10 ± 5.66	60.24 ± 3.36	–	–
RAFNI	87.58 ± 0.19	84.38± 0.56	**79.34 ± 0.51**	88.79 ± 0.19	87.33 ± 0.25	**87.30 ± 1.12**	61.49 ± 1.57	**55.20 ± 1.18**	**45.31 ± 0.96**

used to search for the best hyperparameters for RAFNI. For this proposal, we use ResNet50 in all scenarios, since their method can be used with any CNN.

The authors in [AOA+19] used an original implementation of pre-activation ResNet18 in PyTorch. Unfortunately, we were not able to replicate that network with our algorithm. As a result, we compared our results using RAFNI with our experimental scheme (ResNet50 with fine-tuning, 10 and 15 epochs for CIFAR10 and CIFAR100 respectively) with their model using their experimental scheme (ResNet18, 300 epochs in both cases, data augmentation and learning rate scheduler). In their paper, they also compared their method with other proposals in the literature using different backbone networks.

Finally, we used the same data sets as with our algorithm where it was possible (for the proposals made by [AWA+19] and [MWH+18]), and the given ones in the rest, making sure that the noise injection was the same as it was in our data sets. We argue that, since the noise level is the same, it is introduced randomly in all the cases, and the test sets are also the same as they are predefined, we can safely compare the algorithms.

We also performed a Wilcoxon Rank-Sum test to check if the differences in the results were significant in each case. Since we repeated each experiment five times, we used all five accuracies for each data set and level of noise to perform the Wilcoxon test, instead of using the mean.

In Table 12.10 we show the accuracy obtained with the two approaches proposed in [PRK+17] and our algorithm using the same backbone network and experimental scheme as the one used in [PRK+17]. In the comparison with the estimated backward algorithm, we can see that RAFNI performs better in both data sets at all levels and types of noise. It is interesting that when classifying CIFAR100 with a noise rate of 40% and more, the estimated backward algorithm does not finish the training.

Table 12.11: Comparison between RAFNI and the D2L method [MWH+18], using their original 8-layer CNN for CIFAR10 and pre-activation ResNet44 for CIFAR100 in both approaches. The best results are stressed in bold

	CIFAR10						CIFAR100		
	Symmetric noise			Asymmetric noise			Symmetric noise		
	20%	40%	60%	20%	30%	40%	20%	40%	60%
D2L	86.43 ± 0.15	**84.08 ± 0.30**	**78.32 ± 0.48**	86.75 ± 0.14	85.27 ± 0.27	82.51 ± 0.34	59.18 ± 0.30	28.77 ± 8.52	4.34 ± 1.90
RAFNI	**87.91 ± 0.19**	83.01 ± 0.81	76.37 ± 0.15	**89.04 ± 0.13**	**88.49 ± 0.19**	**86.57 ± 0.17**	**61.49 ± 1.57**	**55.20 ± 1.18**	**45.31 ± 0.96**

Table 12.12: Comparison between RAFNI and the BiTempered method [AWA+19], using ResNet50 in both approaches. The best results are stressed in bold

	CIFAR10						CIFAR100		
	Symmetric noise			Asymmetric noise			Symmetric noise		
	20%	40%	60%	20%	30%	40%	20%	40%	60%
BiTempered	89.33 ± 0.22	76.14 ± 1.12	54.14 ± 1.37	88.82 ± 0.73	84.09 ± 0.63	78.07 ± 0.54	76.51 ± 0.22	71.31 ± 0.32	64.47 ± 0.43
RAFNI	**92.86 ± 0.32**	**89.84 ± 0.66**	**79.48 ± 1.23**	**93.96 ± 0.15**	**92.96 ± 0.13**	**88.51 ± 0.54**	**77.93 ± 0.11**	**73.01 ± 0.16**	**67.08 ± 0.45**

Here, the Wilcoxon-Rank-Sum test obtains that there are significant differences with p-value 9.5×10^{-7}.

When comparing RAFNI with the estimated forward algorithm, we can see that there is less difference between them, especially in CIFAR10, where the differences are usually less than 1% using both types of noise, on average. However, when classifying CIFAR100, RAFNI outperforms the estimated forward algorithm when the noise rate is 40% or more. In particular, RAFNI obtains a gain in accuracy of 6.48% at 40% noise and 11.9% at 60% noise. Using the Wilcoxon Rank-Sum test, we can say that RAFNI performs better with significant differences (p-value 3.6×10^{-5}.

The results we obtained for the D2L algorithm [MWH+18] and our algorithm using the same experimental scheme can be seen in Table 12.11. We can see that for CIFAR10 with symmetric noise there is not much difference between the accuracies obtained by D2L and RAFNI, with D2L being slightly better at 60% noise with a difference of 1.95% on average. But when introducing asymmetric noise, RAFNI obtained better results at all noise rates, with a difference in the accuracy of 4.06% on average at 40% noise. The biggest difference between these two methods, however, can be seen when classifying CIFAR100, where RAFNI outperforms D2L, especially as noise increases, with a difference of 26.43% at 40% noise and 40.97% at 60% noise. The Wilcoxon Rank-Sum test obtains significant differences with p-value 1.31×10^{-5}.

Table 12.13: Comparison between RAFNI, using ResNet50, and the method from Arazo et al [AOA+19], using pre-activation ResNet18. The best results are stressed in bold

	CIFAR10						CIFAR100		
	Symmetric noise			Asymmetric noise			Symmetric noise		
	20%	40%	60%	20%	30%	40%	20%	40%	60%
Arazo et al [AOA+19]	**93.54 ± 0.30**	**92.45 ± 0.18**	**89.34 ± 0.20**	87.60 ± 3.15	80.68 ± 6.46	64.49 ± 27.52	69.29 ± 0.16	63.72 ± 0.19	54.83 ± 0.45
RAFNI	92.86 ± 0.32	89.84 ± 0.66	79.48 ± 1.23	**93.96 ± 0.15**	**92.96 ± 0.13**	**88.51 ± 0.54**	**77.93 ± 0.11**	**73.01 ± 0.16**	**67.08 ± 0.45**

The accuracies we obtained using the BiTempered method and RAFNI, both of them using ResNet50 as the backbone network, can be seen in Table 12.12. Here, RAFNI obtains, on average, better results in both data sets at all noise rates and noise types. The Wilcoxon Rank-Sum obtains significant differences with p-value 1.77×10^{-8}. The biggest differences occur for CIFAR10 with symmetric noise, where RAFNI outperforms Bi-Tempered by 13.7% at 40% noise and by 25.34% at 60% noise.

Finally, the accuracies we obtained with the algorithm proposed by Arazo et al in [AOA+19] and RAFNI, both of them using their original experimental schemes, can be seen in Table 12.13. This is the case where we can see the most discrepancies depending on the data set that is being classified but also depending on the type of noise. For CIFAR10 with symmetric noise, which is easier to classify than CIFAR100, the algorithm from Arazo et al is better than RAFNI, obtaining 9.86% more at 60% noise, on average, though the difference at 20% is considerably lower. However, if we introduce asymmetric noise on CIFAR10, which is a more complicated and realistic type of noise, RAFNI obtains better results. The differences increase as the noise rate increases, being 24.02% at 40% noise. Now, for CIFAR100 with symmetric noise, RAFNI again outperforms the algorithm proposed by Arazo et al, and the differences in accuracy are present even at low percentages of noise, similar to what happened for CIFAR10 with asymmetric noise: RAFNI obtains 8.64% more accuracy, on average, at 20% noise, 9.29% at 40% noise and 12.25% at 60% noise. When we used the Wilcoxon Rank-Sum test on all results, we obtained that RAFNI obtained significant differences with p-value 1.15×10^{-7}.

12.7 COMPARISON WITH AN APPROACH THAT SUPPOSE THE NOISE RATE IS KNOWN

In this section, we compare our algorithm, which assumes no known external information, with the SELFIE algorithm, which assumes the noise rate of the data set

Table 12.14: Comparison between RAFNI and SELFIE [SKL19], using DenseNet-25-12. The best results are stressed in bold

	CIFAR10						CIFAR100		
	Symmetric noise			Asymmetric noise			Symmetric noise		
	20%	40%	60%	20%	30%	40%	20%	40%	60%
SELFIE	**88.04 ± 0.21**	**85.26 ± 0.30**	**77.29 ± 0.45**	**87.90 ± 0.33**	**84.01 ± 0.24**	66.57 ± 0.68	**63.48 ± 0.67**	**60.00 ± 0.40**	**52.62 ± 0.49**
RAFNI	84.00 ± 0.23	78.62 ± 0.84	68.27 ± 0.31	84.48 ± 0.70	82.51 ± 0.65	**78.19 ± 1.83**	56.08 ± 0.33	48.00 ± 0.83	31.91 ± 1.38

is known. It is important to note that, in real scenarios, knowing the noise rate of the data set is not usual. We want to estimate the advantage of using this unusual information in specific models, so that we can quantify the potential loss of not having that kind of knowledge available. Thus, RAFNI is expected to perform worse than SELFIE since RAFNI does not assume extra information, but we want to see in which cases it is competitive.

To make this comparison, we used RAFNI with the same experimental scheme that SELFIE: DenseNet-25-12 as the backbone network, with no data augmentation and the same learning rate scheduler. We train both algorithms for 120 and 150 epochs for CIFAR10 and CIFAR100, respectively, due to time restrictions. For SELFIE, we had to implement the matrix of asymmetric noise as given in [PRK+17], to use the same injection of noise in the data sets used in both algorithms.

The results we obtained for both algorithms are shown in Table 12.14. As expected, SELFIE obtained better results for both data sets and types of noise. Nonetheless, for the CIFAR10 data set RAFNI performs well, obtaining similar results at low noise rates. This also happened when using asymmetric noise, which is interesting since this type of noise is more realistic and difficult to deal with than symmetric noise. In fact, RAFNI outperforms SELFIE at 40% noise by more than 11%.

However, in general, if the information about the noise rate is known, it is better to use an algorithm that uses that information. We did not suppose this type of information was known in our algorithm as we wanted to make it applicable in all cases.

12.8 ANALYSING THE EFFECTIVENESS OF THE RAFNI MECHANISMS

In this section, we analyse how well RAFNI removes and relabels instances using two data sets as samples: EILAT and COVIDGR1.0-SN. In particular, we take a look at a) how many instances were removed, and from those, we check how many of

them were noisy; and b) how many instances were relabelled, and from those, we check how many of them were noisy and how many were correctly classified by the algorithm to their original class.

For both data sets, we analysed how well RAFNI behaves at every level of noise tested in Section 12.5. We used one five-fold cross-validation repeated five times for both data sets, so the results we give here are the total results, that is, for the total number of removals, for example, we sum all the removals in the five training sets. The results for EILAT can be seen in Table 12.15 and the results for COVIDGR1.0-SN can be seen in Table 12.16. In both tables we show 1) the percentage of good removals, that is, instances that were noisy and RAFNI removed from the training set; 2) the total number of instances that RAFNI removed during training; 3) the percentage of good changes, that is, instances that were noisy and RAFNI changed to their original clean class; 4) the total number of instances that RAFNI changed from one class to another during training. Since EILAT has more than two classes, in its case we also show the percentage of noisy changes, that is, instances that were noisy and RAFNI changed to another class, but not their original class.

In both cases, we can see that RAFNI does a good job both removing and changing instances to their original class. In the case of the COVIDGR1.0-SN data set, the percentage of good changes is lower than with EILAT, but this is to be expected since the type of noise introduced in the COVIDGR1.0-SN data set is more difficult and realistic than the symmetric noise introduced in EILAT. Even in that case, RAFNI removes instances that were noisy with an accuracy above 92% at all levels of noise for COVIDGR1.0-SN. On the other hand, RAFNI changes instances to their original class with precision above 95% for EILAT, except at 70% of noise, when it descends to 81.81%. This shows that RAFNI is capable to detect the noisy instances and either remove them or change them to their original class with a high precision, which improves the learning, as we have seen in Section 12.5. The total number of removals and changes tends to increase as the noise level increases in both data sets, as would be expected since the number of noisy instances is increasing, so this is another sign that the algorithm is behaving well.

12.9 CONCLUSIONS

In this paper, we proposed an algorithm, called RAFNI, that can filter and relabel noisy instances during the training process of any convolutional neural network

Table 12.15: Analysis of the instances that the RAFNI algorithm removed and changed from one class to another during the training of the EILAT data set.

Noise	10%	20%	30%	40%	50%	60%	70%
% good removals	75.80%	73.58%	83.58%	88.53%	85.66%	84.80%	82.28%
Total number of removals	2595	4969	6706	9039	18519	13423	15313
% good changes	100%	99.75%	99.22%	98.98%	98.36%	95.63%	81.81%
% noisy changes	0%	0%	0.10%	0.44%	0.44%	2.15%	10.29%
Total number of changes	156	809	1022	683	3176	1486	2303

Table 12.16: Analysis of the instances that the RAFNI algorithm removed and changed from one class to another during the training of the COVIDGR1.0-SN data set.

Noise	10%	20%	30%	40%	50%
% good removals	94.64%	94.15%	94.16%	92.92%	93.27%
Total number of removals	1866	4719	4487	6301	7282
% good changes	40.61%	68.39%	70.88%	74.58%	70.13%
Total number of changes	1091	291	443	535	385

using the predictions and loss values the network gives the instances of the training set. This progressive cleaning of the training set allows the network to improve its generalisation at the end of the training process, improving the results the CNN has on its own. In addition, RAFNI has the advantage that it can be used with any CNN as the backbone network and that transfer learning and data augmentation can be easily applied. It also does not use prior information that is usually not known, like the noise matrix or the noise rate. In addition, it works well even when there is no introduced noise in the data set, so it is safe to use when we do not know the noise rate of a data set. We also made the code available so it is easier to use it.

Developing algorithms that can allow deep neural networks to perform better under label noise is an important task since label noise is a common problem in real-world scenarios and it negatively affects the performance of the networks. We believe that our proposal is a great solution to this problem: it can be easily fine-tuned to every data set, it allows to be used with any CNN, and it allows the use of transfer learning and data augmentation. We proved its potential using various data sets with different characteristics and using three different types of label noise. Finally, we also compared it with several state-of-the-art algorithms, improving their results.

ACKNOWLEDGEMENTS

This publication was supported by the project with reference SOMM17/6110/UGR, granted by the Andalusian Consejería de Conocimiento, Investigación y Universidades and European Regional Development Funds (ERDF). This work was also supported by project PID2020-119478GB-I00 granted by Ministerio de Ciencia, Innovación y Univesidades, and project P18-FR-4961 by Proyectos I+D+i Junta de Andalucia 2018. Anabel Gómez-Ríos was supported by the FPU Programme FPU16/04765 by Ministerio de Educación, Cultura y Deporte.

REFERENCES

[AWA+19] E. Amid, M. K. K. Warmuth, R. Anil and T. Koren, 'Robust bi-tempered logistic loss based on bregman divergences', in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[AOA+19] E. Arazo, D. Ortego, P. Albert, N. O'Connor and K. McGuinness, 'Unsupervised label noise modeling and loss correction', in *International conference on machine learning*, PMLR, 2019, pp. 312–321.

[FV14] B. Frenay and M. Verleysen, 'Classification in the presence of label noise: A survey', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, May 2014.

[GKS17] A. Ghosh, H. Kumar and P. Sastry, 'Robust loss functions under label noise for deep neural networks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[GTL+19a] A. Gómez-Ríos, S. Tabik, J. Luengo, A. Shihavuddin and F. Herrera, 'Coral species identification with texture or structure images using a two-level classifier based on convolutional neural networks', *Knowledge-Based Systems*, vol. 184, p. 104 891, Nov. 2019.

[GTL+19b] A. Gómez-Ríos, S. Tabik, J. Luengo, A. Shihavuddin, B. Krawczyk and F. Herrera, 'Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation', *Expert Systems with Applications*, vol. 118, pp. 315–328, Mar. 2019.

[HZR+16] 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.

[JZL+18] 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', in *International Conference on Machine Learning*, PMLR, 2018, pp. 2304–2313.

[JNC16] I. Jindal, M. Nokleby and X. Chen, 'Learning deep networks from noisy labels with dropout regularization', IEEE, Dec. 2016, pp. 967–972.

[KH+09] A. Krizhevsky, G. Hinton *et al.*, 'Learning multiple layers of features from tiny images', 2009.

[KSH12] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[LHZ+18] K.-H. Lee, X. He, L. Zhang and L. Yang, 'Cleannet: Transfer learning for scalable image classifier training with label noise', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[LWL+17] W. Li, L. Wang, W. Li, E. Agustsson and L. Van Gool, 'Webvision database: Visual learning and understanding from web data', *arXiv preprint arXiv:1708.02862*, 2017.

[MWH+18] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. Erfani, S. Xia, S. Wijewickrema and J. Bailey, 'Dimensionality-driven learning with noisy labels', in *International Conference on Machine Learning*, PMLR, 2018, pp. 3355–3364.

[OTH18] R. Olmos, S. Tabik and F. Herrera, 'Automatic handgun detection alarm in videos using deep learning', *Neurocomputing*, vol. 275, pp. 66–72, Jan. 2018.

[PRK+17] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock and L. Qu, 'Making deep neural networks robust to label noise: A loss correction approach', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1944–1952.

[RLA+14] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan and A. Rabinovich, 'Training deep neural networks on noisy labels with bootstrapping', *arXiv preprint arXiv:1412.6596*, 2014.

[Shi] A. Shihavuddin, *Coral reef dataset, mendeley data, v2*, https://data.mendeley.com/datasets/86y667257h/2, Accesed on: 06-04-2021.

[SC18] G. Song and W. Chai, 'Collaborative learning for deep neural networks', in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, Montréal, Canada: Curran Associates Inc., 2018, pp. 1837–1846.

[SKL19] H. Song, M. Kim and J.-G. Lee, 'Selfie: Refurbishing unclean samples for robust deep learning', in *International Conference on Machine Learning*, PMLR, 2019, pp. 5907–5915.

[SKP+20] H. Song, M. Kim, D. Park, Y. Shin and J.-G. Lee, 'Learning from noisy labels with deep neural networks: A survey', *arXiv preprint arXiv:2007.08199*, 2020.

[SBP+15] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev and R. Fergus, 'Training convolutional networks with noisy labels', 3rd International Conference on Learning Representations, ICLR 2015, Jan. 2015.

[TGM+20] S. Tabik *et al.*, 'Covidgr dataset and covid-sdnet methodology for predicting covid-19 based on chest x-ray images', *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 12, pp. 3595–3605, Dec. 2020.

[WLM+18] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song and S.-T. Xia, 'Iterative learning with open-set noisy labels', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8688–8696.

[XXY+15] T. Xiao, T. Xia, Y. Yang, C. Huang and X. Wang, 'Learning from massive noisy labeled data for image classification', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

[YW19] K. Yi and J. Wu, 'Probabilistic end-to-end noise correction for learning with noisy labels', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7017–7025.

[ZBH+21] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, 'Understanding deep learning (still) requires rethinking generalization', *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, Mar. 2021.

[ZCD+17] H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, 'Mixup: Beyond empirical risk minimization', *arXiv preprint arXiv:1710.09412*, 2017.

[ZS18] Z. Zhang and M. R. Sabuncu, 'Generalized cross entropy loss for training deep neural networks with noisy labels', in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.