FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Deep Learning in Melanoma Detection

Bruno Alexandre Oliveira Dias

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Paulo Reis

Co-Supervisor: Alexandra Oliveira

July 22, 2020

Deep Learning in Melanoma Detection

Bruno Alexandre Oliveira Dias

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Henrique L. Cardoso
External Examiner: Prof. Feliz Gouveia
Supervisor: Prof. Luís Paulo Reis

July 22, 2020

Abstract

Skin cancer is the most common type of cancer, one of the deadliest diseases in the world. However, if detected early, the patient's prognosis is very promising and long term survival rates improve significantly.

Usual diagnostics start with a dermatologist doing a visual examination of a skin lesion that the patient has noticed. There are a few problems with this: 1) The patient needs to have a minimum amount of knowledge to know what to look for, as well as being aware of skin lesions throughout their body. 2) The manual interpretation is time-consuming and prone to human errors. The use of dermatoscopy by dermatologists helps reduce the amount of human errors but it's often not enough.

On another level, recent studies show that applying Machine Learning algorithms to the detection of melanomas can perform on-par and even better than trained dermatologists. Deep learning specifically is becoming more and more used in image classification. Convolutional neural networks are designed specifically for image classification and have been boasting great results in this problem. The ImageNet dataset contains millions of images of thousands of different categories and has been used over the years to develop multiple CNNs with great performance.

Recent studies have been applying CNNs pre-trained on the ImageNet dataset to other image classification problems, including melanoma detection. Combining the feature detection learned by pre-trained CNNs and applying it to a different domain, in a process known as transfer learning, can be extremely useful when trying to classify images using deep learning with a small dataset.

The HAM10000 dataset includes 10015 images of skin lesions, mainly Nevi and Melanomas, and is the largest dataset of its nature publicly available. In this work we train deep learning models based on the VGG16, one of the most famous CNNs created with the ImageNet dataset, using the transfer learning techniques of feature extraction and fine tuning, achieving performance on par with trained dermatologists and even beating them. Further, we will show how different changes to deep learning models affect their performance.

Keywords: skin cancer, melanoma, early diagnosis, Machine Learning, transfer learning, convolutional neural networks, image classification

Resumo

O cancro da pele é um dos tipos mais comuns the cancro, umas das doenças mais letais em todo o mundo. No entanto, quando detetado cedo, o prognóstico do paciente apresenta-se bastante promissor e a taxa de sobrevivência a longo prazo melhora significativamente.

No geral, diagnósticos de cancro de pele começam com a examinação visual de uma lesão na pele que o paciente notou e quis examinar profissionalmente. Existem alguns problemas com isto: 1) O paciente necessita de um mínimo conhecimento do que está à procura e de estar ciente das lesões da pele que tem no seu corpo. 2) A interpretação manual consome bastante tempo e é propensa a erros humanos. O uso de dermatoscopia por dermatologistas ajuda a diminuir esses erros, mas muitas vezes não é suficiente.

Por outro lado, estudos recentes mostram que aplicar algoritmos de Machine Learning na deteção de melanomas consegue atingir performance ao mesmo nível ou ainda melhor do que dermatologistas profissionais. O Deep Learning em específico está a ser cada vez mais usado em classificação de imagens. Convolutional Neural Networks (CNN) são desenhadas especificamente para a classificação de imagens e têm mostrado ótimos resultados neste problema. O dataset ImageNet contém milhões de imagens de milhares de diferentes categorias e tem sido usado ao longo dos anos para criar várias CNNs com alta performance.

Trabalhos recentes têm aplicado CNNs pré-treinadas no ImageNet em outros problemas de classificação de imagens, incluindo a deteção de melanomas. Combinando a deteção automática de caraterísticas aprendida por uma CNN pré-treinada e aplicando-a a um domínio diferente, num processo conhecido como transfer learning, pode ser extremamente útil para a classificação de imagens usando deep learning com um dataset pequeno.

O dataset HAM10000 inclui 10015 imagens de lesões da pele, principalmente de nevos melanocíticos e de melanomas, e é o maior dataset desta natureza publicamente disponível. Neste projeto são treinados modelos de deep learning baseados no VGG16, uma das CNNs criadas com o ImageNet mais famosas, utilizando técnicas de feature extraction e fine tuning, atingindo resultados ao mesmo nível e até acima de dermatologistas profissionais. Para além disso, vamos também mostrar como diferentes mudanças a modelos de deep learning afetam a sua performance.

Palavras-chave: cancro da pele, melanoma, diagnóstico precoce, Machine Learning, transfer learning, convolutional neural networks, classificação de imagens

Acknowledgements

I would like to take this opportunity to thank my supervisor Luís Paulo Reis and especially my co-supervisor Alexandra Oliveira for the support during the development of this dissertation. Their help was very important in particular during the final stages of writing and during a change in the topic of the dissertation.

The greatest thank you goes to my family and especially my parents for their continued support over the years and for always putting education first in their list of priorities.

A final thank you goes to all the friends I've made over five years in Porto. All the experiences we've lived will forever be on my memory and I hope we all stay in touch.

Bruno Dias

"The greatest glory in living lies not in never falling,
but in rising every time we fall."

Nelson Mandela

Contents

List of Figures

List of Tables

Abbreviations

ANN	*Artificial Neural Network*
AdaGrad	*Adaptive Gradient Algorithm*
Adam	*Adaptive moments*
BCC	*Basal Cell Carcinoma*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DT	*Decision Tree*
FC	*Fully connected*
GAP	*Global Averaging Pool*
GPU	*Graphics Processing Unit*
ML	*Machine Learning*
MSE	*Mean Square Error*
RMSProp	*Root Mean Square Propagation*
ReLU	*Rectified Linear Unit*
ResNet	*Residual Network*
SCC	*Squamous cell carcinoma*
SGD	*Stochastic Gradient Descent*
SOTA	*State of the Art*
SVM	*Support-Vector Machine*
k-NN	*k-Nearest Neighbor*

Chapter 1

Introduction

1.1 Context

Skin cancer is the most common type of cancer, one of the deadliest diseases in the world. If detected early, the patient's prognosis is very promising and long-term survival rates can be improved significantly [36].

As technology progresses and its' role in society becomes more and more important, its use in the population's health and well-being becomes imperative. Our health is one of our most important assets. To help keep it, technology can be used to assist medical professionals in their jobs, helping them use their time more efficiently and effectively. In order to understand how technology can be of use, first we need to understand how skin cancer presents itself and how it is traditionally diagnosed.

1.2 Motivation

The human skin is the largest organ of the human body and covers its' outer area, protecting it from microbes and the elements. The skin also helps regulate body temperature and permits the sensations of touch, heat, and cold [32].

According to Millington in the book *Skin*, the skin is composed of mainly three layers [50] (illustrated in figure 1.1):

- The **epidermis** is the outermost layer. It provides a waterproof barrier and is responsible for the skin tone.
- The **dermis** is the layer beneath the epidermis and contains tough connective tissue, hair follicles and sweat glands.
- The **hypodermis** is a deeper subcutaneous tissue that is mainly made of fat and connective tissue.

As any living organ, the skin can be affected by lesions that have varying degrees of severity and different consequences. A skin lesion is a part of the skin that has an abnormal growth or

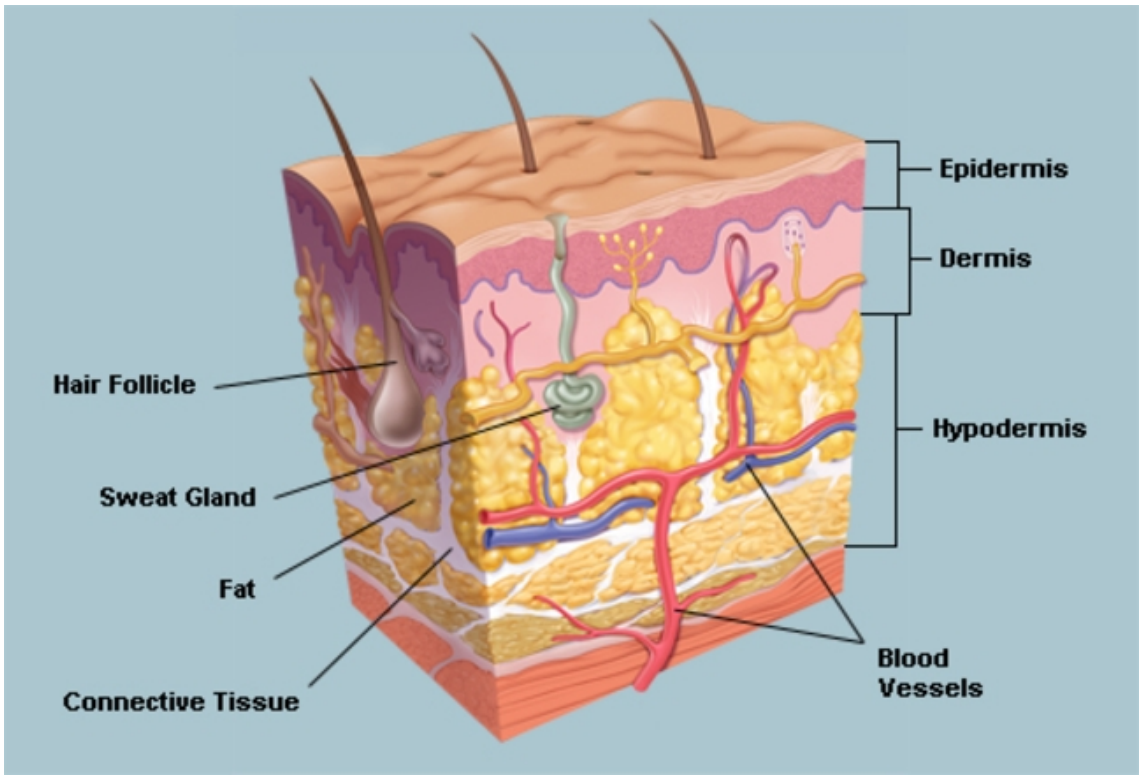


Figure 1.1: Human Skin Anatomy. Obtained from [32].

appearance compared to the skin around it [33]. There are many different types of skin lesions, ranging from freckles or other benign conditions to more serious problems, such as skin cancers.

There are three main types of skin cancer: basal cell carcinomas, squamous cell carcinomas and melanomas [36]. Apart from those, melanocytic nevi present their own clinical interest as they have the possibility of evolving into melanomas, despite not being cancerous themselves [72]. Distinction between types of skin cancer is mainly done by assessing different shapes, colors or textures. The appearance of complications such as ulcerations and the location of lesion can also be used as differentiating factors.

Basal cell carcinoma (BCC)

Although there are many clinical presentations of BCC, the most common characteristic type is a raised, pearly bump which can be ulcerative or asymptomatic. Sometimes small blood vessels (telangiectatic vessels) can be seen on the tumor [36].

This cancer type appears mostly in sun-exposed skin of the head, neck, torso and shoulders. High-risk areas for tumor recurrence after initial treatment include the central face (e.g., periorbital region, eyelids, nasolabial fold, or nose-cheek angle), postauricular region, pinna, ear canal, forehead, and scalp [36].

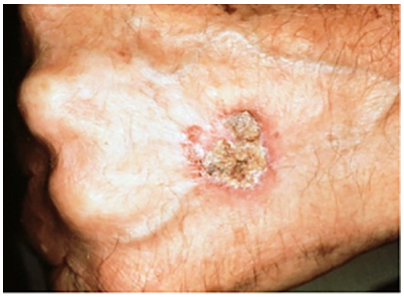
(a) BCC as a shiny bump or nodule

(b) BCC as an open sore that does not heal

Figure 1.2: Photos of two different appearances of basal cell carcinomas. Obtained from [41].

Squamous cell carcinoma (SCC)

SCCs tend to present as a red, scaling, thickened patch, open sores or wart-like skin on sun-exposed portions of the skin, such as the ears, lower lip, and dorsa of the hands. They are composed of keratinizing cells, some are firm hard nodules and dome shaped like keratoacanthomas, and ulceration and bleeding may occur. If left untreated, SCCs may develop into a large mass [36].



(a) SCC as a persistent, scaly red patch

(b) SCC as a wart-like growth that crusts and sometimes bleeds

Figure 1.3: Photos of two different appearances of squamous cell carcinomas. Obtained from [27].

Melanoma

Melanoma is a type of cancer that begins in melanocytes (cells that make the pigment melanin). Although it is most common in the skin, it can also start in the eye, the intestines, or other areas of the body with pigmented tissues [37].

Melanomas are identified by mainly five characteristics. The mnemonic 'ABCDE' helps to remember to look for a lesion's asymmetry, borders, color, diameter and evolution (illustrated in figure 1.4):

- **Asymmetry**: The shape of one half of the lesion is different from the other.

- **Border**: Borders are irregular, the edges are often ragged, notched, or blurred in outline. The pigment may spread into the surrounding skin.
- **Color**: More than one color, with darker or variable discoloration. Shades of black, brown, and tan may be present. Areas of white, gray, red, pink, or blue may also be seen.
- **Diameter**: Usually above 6mm.
- **Evolution**: The mole changed in the past weeks or months. Changes can be in size, shape, color or feel. [37]

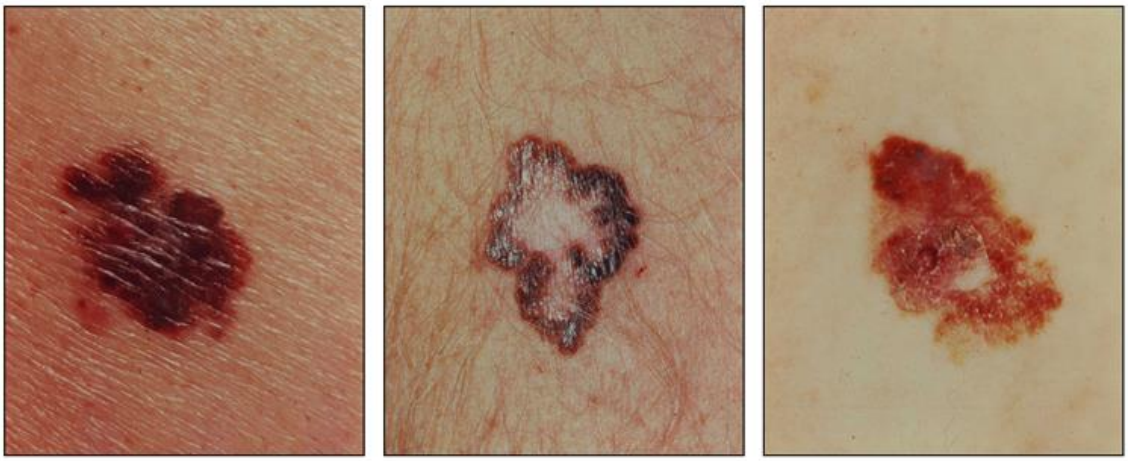


Figure 1.4: Examples of melanomas with characteristic asymmetry, border irregularity, color variation, and large diameter. Obtained from [35].

Melanocytic Nevus

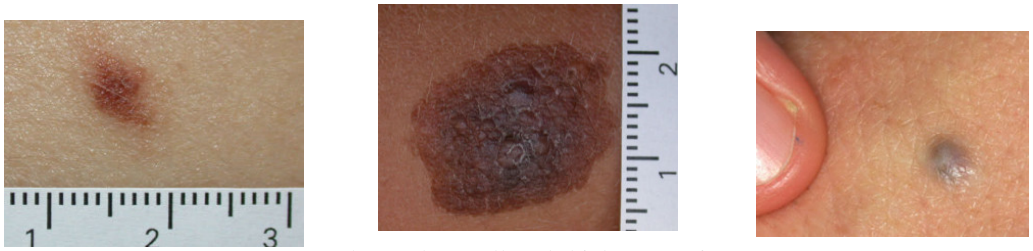
Commonly referred to as a mole, a melanocytic nevus is a type of melanocytic tumor that contains nevus cells [20]. They are commonly congenital or appear in childhood and teenage years [72].

Although typical moles consist of a simple brown spot, they can pose different characteristics:

- **Color and texture**: Moles can be brown, tan, black, red, blue or pink. They can be smooth, wrinkled, flat or raised. They may have hair growing from them.
- **Shape**: Most moles are oval or round.
- **Size**: Usually smaller than 6mm, although congenital moles can be much bigger [72].

Melanocytic nevi are **benign** moles. Over time, however, these can change appearance and evolve into melanomas [72]. As about one third of melanomas derive from melanocytic nevi [17], it's important to be able to distinguish the two, despite their similar appearance. The ideal scenario would be to reliably predict which nevi will evolve into melanoma and treat them before said evolution. Very often the patient is the one responsible for noting a lesion's evolution and

communicating it to a health professional, which makes it difficult for there to exist data recording the evolution of melanocytic nevi which could be used in an attempt to predict their evolution into melanomas [69]. Until standards change and historical data, containing different stages of nevi becoming melanoma, is recorded and made available, distinction between the two lesions is mostly done using symmetry, border irregularity and color characteristics.



(a) Small, brown and flat nevus

(b) Dark, small and thick congenital nevus

(c) Blue nevus

Figure 1.5: Photos of three different appearances of melanocytic nevi. Obtained from [19].

As the human population increases, there are more and more people recurring to hospitals and health clinics to see medical professionals.

Early diagnosis of certain conditions, such as some types of skin cancer, is mandatory as they can quickly become fatal. With the increase of the population, it can be difficult to have enough manpower to professionally and individually test every person regularly for signs of the early stages of skin cancer. Further, even professional dermatologists can't diagnose these cancers with absolute certainty without recurring to invasive biopsies. Brinker [8] conducted a survey in which german dermatologists were asked to distinguish melanoma and nevi images out of different datasets of 100 dermoscopic or clinical images. This study shows a group of 157 dermatologists scoring an overall sensitivity of 74.1% and specificity of 60.0% using images obtained through dermoscopy tests and a group of 145 dermatologists scored an overall sensitivity of 89.4% and specificity of 64.4% using non-dermoscopic images.

Technology, specifically machine learning, can be helpful by providing reliable and accessible systems that are able to point the potential diagnostics, whether they will be used by the general population in self-examination or by dermatologists to help increase the reliability of diagnoses and hopefully reduce the need of recurring to biopsies when unnecessary.

1.3 Objectives

The objective of this dissertation is to create a machine learning system capable of classifying images as melanoma or melanocytic nevi with equal or higher scores than those achieved by dermatologists, while using the biggest publicly available skin lesion dataset, with reliable ground-truth labels, and comparing different approaches to image pre-processing, feature extraction and classification. Another objective is the analysis of the impact of changes to transfer learning classification models in a binary classification process.

1.4 Document Structure

This document is structured in five chapters. After finishing this introduction, chapter two will focus on the state of the art, specifically of machine learning algorithms, deep learning and learning techniques, model performance metrics and a review of related work and datasets. Chapter three will hold a description of the methodology used in building and testing different methods. Chapter four will hold the results of relevant tests. Finally, chapter five will hold the conclusions and future work.

Chapter 2

State Of the Art

Several methods have been introduced to classify images and specifically melanoma. These methods range from traditional machine learning algorithms such as k-Nearest Neighbors algorithms (k-NN) and linear support-vector machines (SVMs) to deep learning approaches using convolutional neural networks (CNNs).

This chapter will start by introducing the basics of machine learning and some of the traditional algorithms. This is followed by a description of artificial and convolutional neural networks and the use of transfer learning to quickly train new models based on pre - trained networks, using little amounts of data. After, we will introduce widely used performance metrics used to evaluate these algorithms. Finally, we will preview related datasets and works that share similarities with the proposed in this dissertation.

2.1 Machine Learning

Arthur Samuel defines Machine learning (ML) as the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead [44].

Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [52].

Machine Learning algorithms can be divided into three basic types, depending on the learning problem:

- **Supervised learning** is the task of learning a function that maps an input to an output based on example input-output pairs given. A number of N inputs is given to the algorithm where the output for each N input is known beforehand. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

- **Unsupervised learning** is a type of learning that helps find previously unknown patterns in a data set without pre-existing labels. A number of N inputs is given to the algorithm where the output for each N input is not known beforehand. This allows the modeling of probability densities functions of given inputs.

- **Reinforcement learning** is learning how to map situations to actions so as to maximize a numerical reward signal. The algorithm is not told which actions to take, but instead must discover which actions yield the most reward by trying them [73]. The algorithm is given a set of states about the environment and a set of actions, with an associated reward value, that can be taken to transition between states. With the given information, the algorithm must find the best actions to take for a specific environment.

Machine Learning algorithms can also be divided into three categories, depending on the output:

- **Classification** algorithms want to predict the class label output of an input (e.g. predicting if e-mails are spam or not).

- **Regression** algorithms want to predict a continuous quantity output of an input (e.g. predicting houses' selling price).

- **Clustering** algorithms group data according to their degree of similarity.

There are multiple machine learning algorithms already developed that can be used for various specific tasks. Despite not being originally designed with image classification in mind, some of these algorithms can be adapted to classify images, especially when using a different algorithm to automate feature extraction. Decision trees, random forests, k nearest neighbor and support-vector machines are some of the algorithms that can be used in this scenario.

These algorithms are still sometimes used in image classification but have been falling out of favour with the rise in interest of deep learning (a subset of machine learning) algorithms, particularly convolutional neural networks, which are specifically designed for image classification. Deep learning concepts are later expanded in section 2.2.

2.1.1 Decision Tree

Decision Tree (DT) is a supervised learning algorithm that can be used to solve both classification and regression problems. It takes the form of a tree structure with **decision nodes** and **leaf nodes**. A decision node has two or more branches. Leaf node represents a class label in the case of classification problems or a continuous value (typically a real number) in the case of regression problems. There are multiple algorithms that implement decision trees already developed and are widely used, such as ID3, C4.5 and CART (Classification and Regression Trees).

The Iterative Dichotomiser 3 (ID3) algorithm is mostly used for classification problems with discrete data and uses a top-down greedy approach to build a decision tree starting at the top and always choosing the best feature at the moment. The C4.5 algorithm is an extension to ID3 by the

same author and shows some improvements such as the handling of both discrete and continuous data and pruning trees after creation by removing branches that don't contribute significantly to the decision process. The CART algorithm can handle both classification and regression problems and uses a new metric named gini impurity 2.1, which measures how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled, to create decision points for classification tasks [5, 57, 58, 67].

$$Gini = 1 - \sum_{i=1}^{N} p_i^2 \tag{2.1}$$

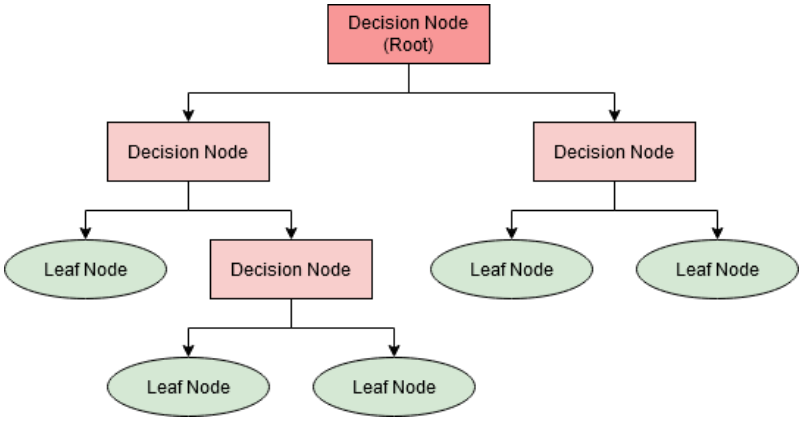


Figure 2.1: Illustration of the general appearance of a simple decision tree.

2.1.2 Random Forest

A **random forest** is a supervised learning method that can be used for classification or regression problems. In essence it consists of an ensemble of a multitude of different decision trees where each individual tree predicts a class and the most common prediction is chosen as correct. Rather than having a single very complex tree, which tends to overfit, this algorithm uses multiple different trees, each sensitive to different feature dimensions [31].

Deep forest, or gcForest, is an ensemble of random forests in a cascading structure which is said to enable gcForest to do feature extraction in a similar way to deep neural networks [78].

2.1.3 k-Nearest Neighbor

The k-nearest neighbors algorithm (k-NN) is a non-parametric method used to solve both classification and regression problems [2]. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- **Classification:** the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k

nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

- **Regression:** the output is the property value for the object. This value is the average of the values of k nearest neighbors.

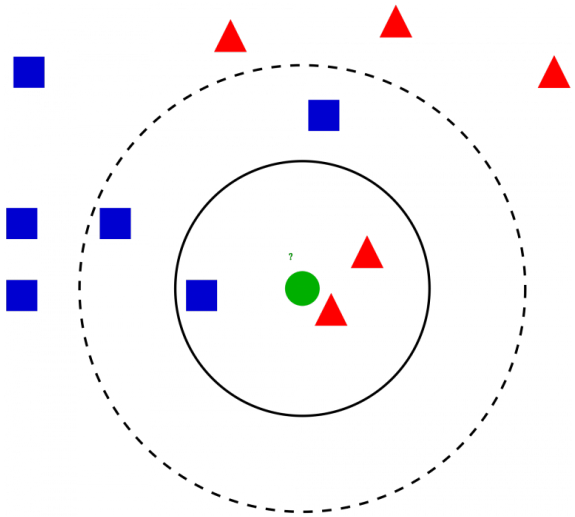


Figure 2.2: Visualization of k-NN algorithm. In this example, if *k* is 3 (inner circle) the test sample (green circle) is classified as a red triangle since there are 2 red triangle and 3 blue squares. If *k* is 5 (outer, dotted circle) the test sample is classified as a blue square (3 blue squares vs 2 red triangles). Image obtained from [71].

2.1.4 Support-Vector Machine (SVM)

A support-vector machine is a supervised machine learning model that uses classification algorithms for two-group classification problems. In general, a SVM maps input vectors non-linearly to a very high-dimension feature space, where a linear decision surface, or hyperplane, is constructed [16]. As seen in figure 2.3, multiple hyperplanes can separate the data and, intuitively, the best hyperplane is the one where the distance from it to the nearest data point on each side is maximized. In general, to the increase in the margin between the hyperplane and nearest data points corresponds a decrease in the generalization error of the classifier [29].

2.2 Deep Learning

The concept of Deep Learning can be traced back to 1943 and the creation of a computer model based on the neural networks of the human brain by Walter Pitts and Warren McCulloch. In their seminal work "A Logical Calculus of Ideas Immanent in Nervous Activity" [48], they proposed a combination of algorithms and mathematics they called “threshold logic” with the aim of mimicking human thought process. Having evolved over the years, their model remains a standard in current days.

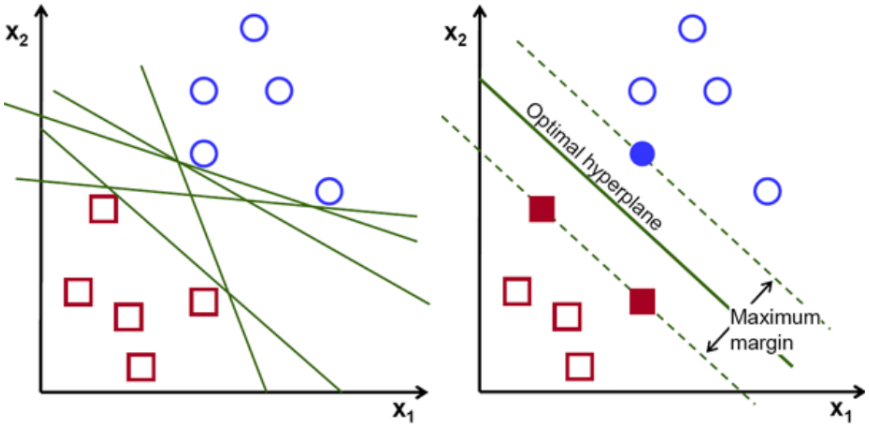


Figure 2.3: Illustration of multiple hyperplanes separating two data groups and the choice of an optimal hyperplane. Source: [39].

In 1958, Frank Rosenblatt published "The perceptron: a probabilistic model for information storage and organization in the brain" [62] introducing the percepton, capable of automatically learning a mapping from real-valued inputs to a binary output. Having only a single layer, perceptrons are only capable of learning linearly separable patterns. In 1969 Marvin Minsky and Seymour Papert published the famous book "Perceptrons" [51], which showed that networks of perceptrons were incapable of learning the XOR operation, a non-linear function.

Over the years deep learning kept slowly evolving. In 1965 the first working deep learning networks were created by Alexey Ivakhnenko and V.G. Lapa [40] with learning algorithms that used deep feedforward multilayer perceptrons and in 1979 Kunihiko Fukushima led the way to convolutional neural networks with the development of Neocognitron, an artificial neural network which used a hierarchical, multilayered design that allowed the computer to “learn” to recognize visual patterns [24].

Possibly the most influential work was done by Rumelhart, Williams, and Hinton, with their published research [64] describing in great detail the process of backpropagation and how it could vastly improve the existing neural networks. To this day, backpropagation remains widely used as a learning mechanism in all modern deep learning applications.

In more recent years, deep learning research has been fueled by the continuous rise in large scale datasets as well as the increase in capability and availability of GPUs. The ImageNet dataset, in particular, is part of the ImageNet Large Scale Visual Recognition Challenge, a benchmark in object category classification and detection. It has more than 14 million hand-annotated images of more than 20 thousand different categories, such as different types of foods, flowers, animals, fungus, structures and many others, and has allowed the creation of multiple deep learning models with increasing performance and accuracy that can later be transferred to other domains [18, 65].

2.2.1 Concepts and Techniques

There are some important principles and techniques about deep learning worth mentioning in the scope of this dissertation:

- **Tensor:** A tensor is "an array of numbers arranged on a regular grid with a variable number of axes" [25]. It can be easily understood as a multidimensional array and is the primary data structure used in deep learning.

- **Activation Function:** Activation functions are mathematical equations that determine the output of each neuron in a neural network, determining whether it should be activated ("fired") or not, based on their input. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

- **Feature Map:** A feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. This passes through an activation function, forming a tensor that is then used as an input for the next layer.

- **Loss function:** A loss function serves as a measure of how good a model is at predicting the expected outcome. An optimization problem seeks to minimize loss by adjusting parameter values in a neural network model. There are multiple different loss functions that work better in different kinds of data. The most common loss functions used in neural networks are the Mean Square Error (MSE) and the Cross Entropy Loss. The former is preferred for regression while the latter is one of the best choices for classification and can be calculated as shown in equation 2.2. In this equation M is the number of classes, $y_{o,c}$ is binary indicator (0 or 1) if class label c is the correct classification for observation o, $p_{o,c}$ is the predicted probability observation o belongs to class c.

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{2.2}$$

- **Gradient:** An error gradient is the direction and magnitude calculated during the training of a neural network that is used to update the network weights.

2.2.2 Neural Networks

An **artificial neural network** (ANN) can be either a supervised learning algorithm, used for classification or regression problems, or an unsupervised learning algorithm.

The network itself is a multi-layered graph, with three main types of layers:

- The **input layer** where initial data is entered.

- The **hidden layers** are intermediate layers used for intermediate computations. The number of intermediate layers and nodes used affects the performance of the algorithm.

- The **output layer** where the output or prediction can be obtained and where the output data is entered if the algorithm is supervised.

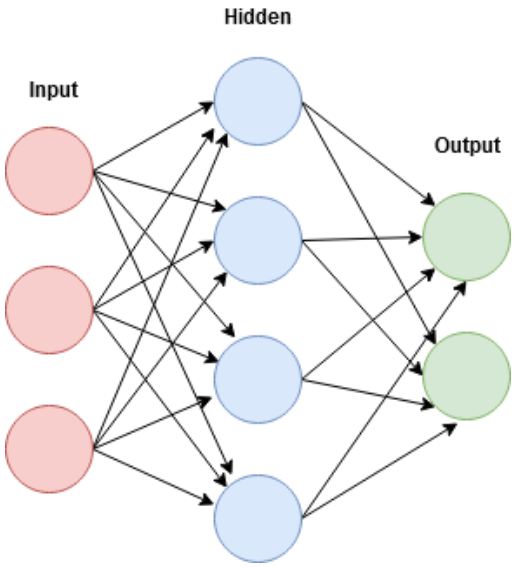


Figure 2.4: Illustration of an artificial neural network.

An ANN with multiple hidden layers is called a deep neural network (DNN). The convolutional neural network (CNN) is a class of DNNs most commonly applied to analyzing visual images.

CNNs have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing [54, 15]. They use relatively little pre-processing compared to other more traditional image classification algorithms. This is possible because the network attemps to learn by itself high level features from massive amount of data, which in traditional algorithms is hand-engineered. This automatic extraction of data features reduces the need of domain expertise for feature engineering, reducing the burden on users. This is particularly important in health as it is a particularly complex area that has direct implications in human life.

Central to the CNN is the convolutional layer that gives the network its name. This layer performs an operation called a "convolution", illustrated in figure 2.5. A convolution is a linear operation that involves the multiplication of a set of weights with the input. As shown in equation 2.3, with 2D input the multiplication is performed between the input data I and a smaller two-dimensional array of weights K, called a filter or a kernel. The multiplication used is a dot product, an element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value $C[m,n]$. In this equation m and n are the positions of the result in the feature map output and u and v are the width and height of the kernel, used to

iteratively multiply different values on the input with different values on the kernel. After each convolution operation, a bias value associated with each kernel is added to the result.

Using a filter smaller than the input allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input, in each overlapping part of filter-sized patch of the input data, left to right and top to bottom. This systematic application of the same filter across an input can give the model some degree of translation invariance, if the filter can detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the detection of that feature anywhere in the image.

$$C[m,n] = \sum_u \sum_v I[m+u, n+v] * K[u,v] \tag{2.3}$$

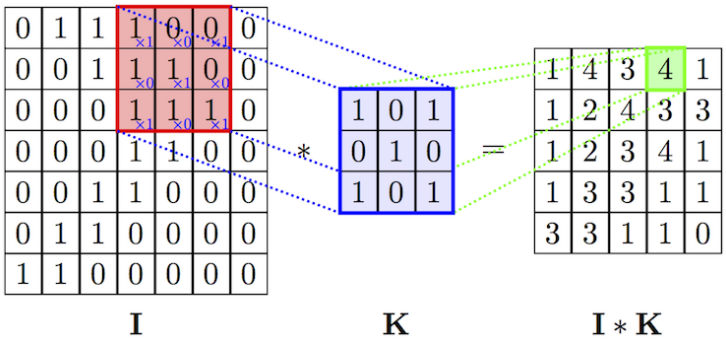


Figure 2.5: Illustration of the dot product operation done in a convolution.

Convolutional neural networks (illustrated in figure 2.6) take advantage of the input being images, which have an inherent grid-like structure, and conform their architecture to arrange image data in 3 dimensions: width, heigh and depth. After several layers of transformation through differentiable functions the output is reduced from a full image to a single vector of class scores. The differentiable functions used in CNNs are typically convolutions followed by activation functions, such as ReLU, sigmoid or hyperbolic tangent, followed by average or max pooling, which reduces the spacial size of the representation by summarizing the average or most activated, respectively, presence of a feature in patches of a feature map.

As seen in figure 2.6, CNNs have a final section that handles classification. There are three main ways to set up this section, with different layers, but first it is important to define a few concepts:

- A fully connected (FC) layer connects every neuron in one layer to every neuron in another layer and performs the operation $output = activation(dot(input, kernel) + bias)$ where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer.

- The dropout layer was introduced as a regularization method to reduce overfitting and improve generalization error. It works by randomly ignoring or 'dropping out' some nodes

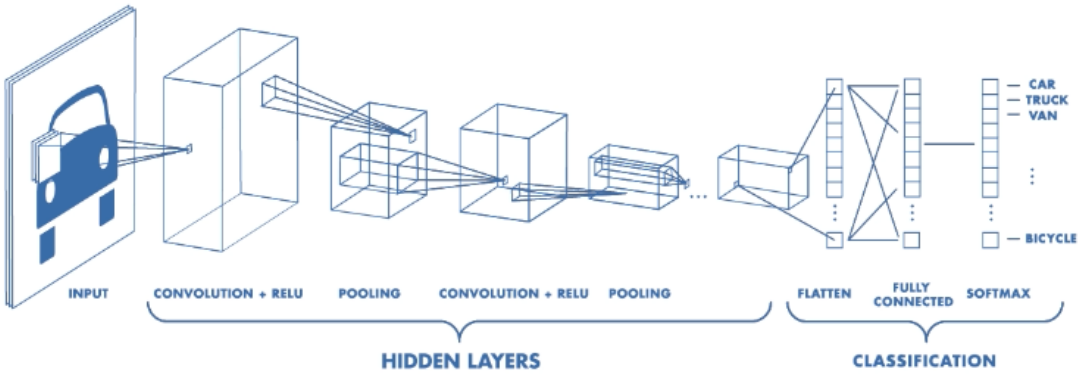


Figure 2.6: Illustration of a typical CNN. Obtained from [10].

(and their connections) from the network during training, which prevents nodes from co-adapting too much and reduces overfitting [70].

- The BatchNormalization layer was introduced in [38] and aims to improve speed, performance and stability of neural networks by normalizing the input layer and re-scaling.

- The Rectified Linear Unit (ReLU) activation function f outputs the input directly if is positive, otherwise, it will output zero. ReLU is preferred over other activation functions such as hyperbolic tangent or sigmoid due to considerably faster training. [45].

$$f(x) = max(0;x) \tag{2.4}$$

- A global average pooling (GAP) layer calculates the average output of each feature map in the previous layer, down sampling each entire feature map to a single value [46]. It can be used in a model to aggressively summarize the presence of a feature in an image and was initially proposed in [46] to replace the traditional fully connected layers in CNN.

- Softmax is a multi-class generalization of the sigmoid activation. It takes as input a vector x of K real numbers, where K is the number of classes in the problem, and normalizes it into $[0,1]$ values proportional to the exponentials of the input numbers. The result is a vector with each value in the interval $[0,1]$ and the sum of all values will be 1, so it can be interpreted as probabilities. This is usually used as an activation function in the final fully connected layer with a number of nodes equal to the number of classes.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \text{ where } x_i \text{ is the } i\text{th element of the set of input values.} \tag{2.5}$$

The 'traditional' way to handle classification consists of adding a flatten layer that turns the output into a single feature vector and then adding a variable number of fully connected (FC) layers, usually two is enough to combine the features detected with the previous layers, with ReLU activation and finally a FC layer with softmax activation. The first fully connected layers can have a variable number of nodes, while the softmax activated one must have exactly as many nodes as there are classes. Since the introduction of dropout layers by Sristava in [70], they are also often used in between FC layers to mitigate overfitting. Batch normalization layers serve a similar purpose and are added before classification.

This traditional approach is very common and provides reliable performance. A notable example of its use is the original VGG16 model [68] which, after flattening the output into a single feature vector, used two ReLU activated FC layers with 4096 nodes each followed by a softmax activated FC layer with 1000 nodes to handle the classification of 1000 classes.

Lin in [46] later proposed a new way to handle classification with the creation of global averaging pooling (GAP) to replace the fully connected layers. Having one feature map for each class of the classification task, instead of adding FC layers, a GAP layer is used to take the average of each feature map and feeds the resulting vector directly into the softmax layer. Since there are no parameters to optimize in the global average pooling, overfitting is automatically avoided at this layer [46]. This approach only works if the feature extraction layers are prepared to output as many feature maps as the number of different classes.

The third approach was used by the original ResNet50 model [30] and can be considered a middle-ground between the other two (GAP + FC). In this model, the GAP layer is followed by one fully connected layer with a softmax activation function that yields the predicted object classes. Unlike the previous approach, this does not require the output of a specific number of feature maps as the final fully connected layer combines features in as many nodes as there are classes to predict.

VGG16 [68] and ResNet50 [30] are two commonly used state-of-the-art CNN's that were pre-trained on the ImageNet dataset [61]. The VGG and ResNet architectures won the ImageNet Large Scale Visual Recognition Challenge, from which the dataset originates, in 2014 and 2015, respectively.

Back in 2014, the creation of the VGG16 and VGG19 architectures, with small and fixed sized kernels in each layer, kicked off the exploration of deeper neural networks, which until then, were found difficult to train efficiently and actually lost effectiveness with the increase in layers. Despite drawbacks when compared to newer CNNs, such as the slower training and large network architecture weights, VGG16 is still a popular architecture, particularly for learning purposes due to its' simplicity [68, 28].

The VGG16 architecture (visualized in figure 2.8) receives inputs of size 224x224x3, the filters used have a size of 3x3 and max pooling is done in groups of 2x2 pixels, halving the spatial resolution of feature maps in width and height. At the top of the network (the last layers) classification is done using two fully connected layers with 4096 nodes each and a final softmax layer

with 1000 nodes, representing 1000 classes. All hidden layers, convolution and fully connected alike, are equipped with ReLU activation [68].

The exact configuration can be seen in figure 2.7. Reading column D, which represents the VGG16 architecture, the first convolutional layer has to learn 64 filters of size 3x3 along the input depth of 3. Since each filter also has its own bias this gives a total number of $64 * 3 * 3 * 3 + 64 = 1792$ parameters. The same logic can be used to calculate the parameters for the remaining convolution layers. The filters used are designed to preserve spatial resolution, by having padding and stride set to 1. Padding adds extra pixels set to 0 all around the image, allowing the filter to be used on the edges of the image, and the stride is the number of pixels advanced between each convolution - having it set to 1 means no pixels are skipped. Due to this, only pooling layers change the spatial resolution, making the output of the first convolution layer is 224x224x64 and the output of the first pooling layer 112x112x64.

The number of parameters in fully connected layers is calculated by multiplying the number of nodes in the previous layer with the number of nodes in the current layer. Following previous logic, the output of the last layer before the first fully connected layer has a size of 7x7x512. The number of parameters in the first fully connected layer is then $7 * 7 * 512 * 4096 + 4096 = 102764544$.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.7: ConvNet configurations. Column D and E represent the VGG16 and VGG19 architectures, respectively. Obtained from [68].

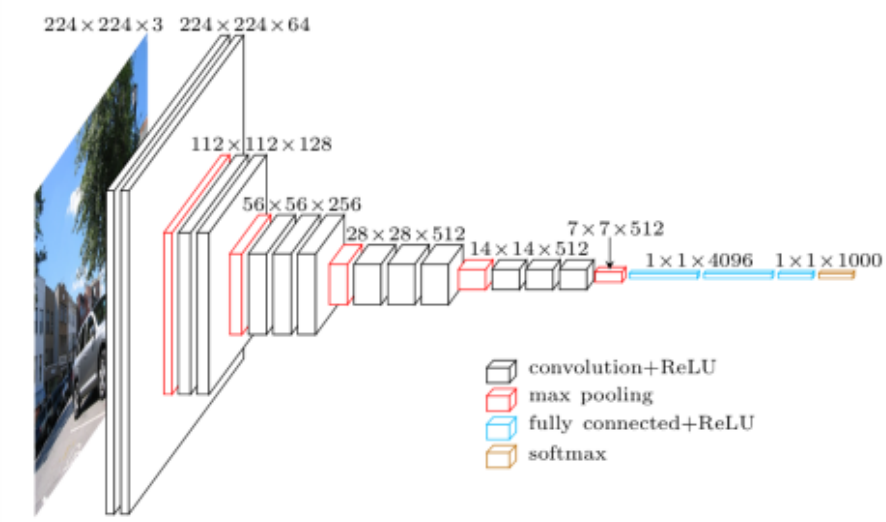


Figure 2.8: Visualization of VGG16's architecture. Obtained from [23].

In 2015, it was time for the ResNet architecture to innovate. A residual neural network (ResNet) can afford to be much deeper than previous architectures by resolving the vanishing gradient problem. Previously, it was noted that deeper networks would have higher training error than shallower networks as the gradients from where the loss function is calculated easily shrink to zero. With ResNets, the gradients can flow through the skip connections backwards from later layers to initial filters [30].

The ResNet50 architecture is much more complex than the VGG. It receives inputs with the same size of 224x224x3 but has varying filter size. Analysing figure 2.9, it can be seen that it starts with a convolutional layer with 64 7x7 filters followed by a 3x3 max pool layer, both with a stride of 2, which means both layers halve the size of the feature maps. Following are multiple different building blocks shown in brackets, the first includes three convolutional layers with 64 1x1 filters, 64 3x3 filters and 256 1x1 filters, respectively, and is repeated three times, giving a total of 9 layers. After the first group of building blocks, the first layer on the first repetition of each block has a stride of 2, halving the feature map size before the more computationally expensive layer with 3x3 filters. The first two layers of each block use ReLU activation and batch normalization is used after every convolutional layer. The final layers are average pooling followed by a fully connected layer with 1000 nodes and softmax activation. Using skip connections (visualized in fig 2.10), after each repetition of a building block the output is sent as input not only to the following building block but also to the building block after that, effectively skipping 3 convolutional layers.

2.3 Transfer Learning

The main problem in deep learning is data dependence [74]. While these algorithms require very large amounts of data to understand existing patterns and provide good enough predictions, many real world problems just don't have that much data available. However, using pre-trained convolutional neural networks, such as the VGG16 or ResNet50 mentioned above, it is possible to use

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.9: ResNet architectures. Column with 50 layers represents the ResNet50. Obtained from [30].

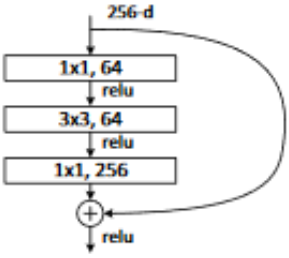


Figure 2.10: ResNet skip connection. Obtained from [30].

deep learning with comparatively little data. This is possible due to the ability to transfer knowledge gained by the pre-trained model to a different but related problem through a process known as *transfer learning*. Given the very high variety of classes in the ImageNet dataset, networks trained on it are easily capable of detecting features that can often be translated to other classes not originally present.

Transfer learning has the following definition according to Pan et al. in [55]:

Definition 1 *Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to improve the learning of the target predictive function $f_T(.)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.*

In this definition, a domain $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution *P(X)*, where X = $\{x_1,...,x_n\} \in \mathcal{X}$. Given a specific domain, $\mathcal{D}$=$\{\mathcal{X}$,P(X)$\}$, a task consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$.

Tan et al. in [74] distinguish four different categories of deep transfer learning: instances-based, mapping-based, network-based and adversarial-based. This work will focus on the network-based category, which is commonly used in image classification problems. In practice, this works by using a pre-trained network, without the last layers that perform the classification but with the pre-trained weights of each of the previous layers (i.e. the 'knowledge' learned by the network), as a basis to create a new model. In the case of image classification problems, networks trained on the *imagenet* database, such as the VGG16 and ResNet50 described earlier, are good choices for network-based transfer learning. Figure 2.11 illustrates the general idea of network-based transfer learning.

Using the concept of freezing layers in a CNN, which consists of disabling training on a layer so that its weights are no longer updated, transfer learning can be divided in two different approaches known as feature extraction and fine tuning. It's important to notice that the expressions *fine tuning* and *transfer learning* are sometimes used interchangeably but the former is also referred to as an approach to the latter. When using a pre-trained model to classify images that are considerably different from the ones used to train it, it's expected that fine tuning will perform better than just using a feature extraction approach. This is due to the model not being prepared to detect new features that only exist in these new images, which can be improved by fine tuning the convolutional layers.

2.3.1 Feature Extraction

After removing the top (the last layers that are responsible for classification) of a pre-trained convolutional neural network it can then be used as a arbitrary feature extractor. This is done by simply allowing the input to forward propagate until the last layer and retrieving its output, which would normally be fed as input to the classifier, as the extracted features. This output differs based on which base model was used. For example, the VGG16 model's output has the volume shape of $7 * 7 * 512$ [68], which can be flattened to a feature vector of dimension 25088, while the

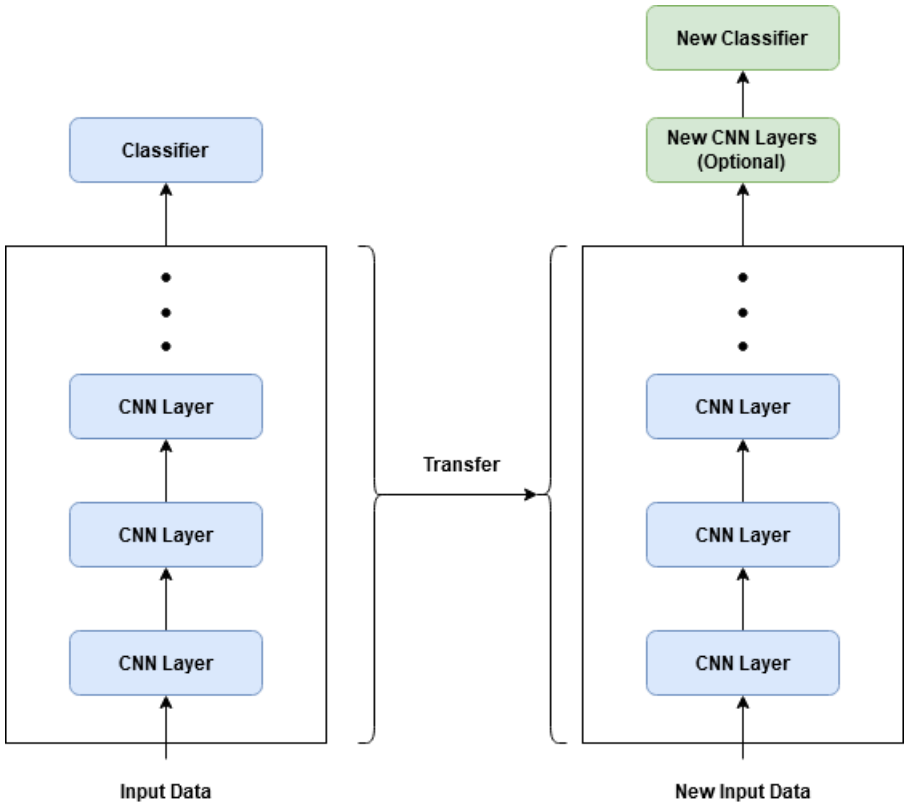


Figure 2.11: Illustration of a general example of transfer learning. The left represents the pre-trained network which, without the classifier, gets used as a base on the network to the right, transferring the knowledge obtained initially.

ResNet50 model has output in the volume shape of $7 * 7 * 2048$ [30], resulting in a feature vector of dimension 100352.

The feature extraction process is repeated for every image of a dataset, resulting in N feature vectors, where N is the number of images in the dataset. The vectors can then be used to train a traditional machine learning model, which comes with memory-related problems if the dataset has a considerable size. *Logistic regression* and *linear SVM* are commonly used traditional models in the context of feature extraction due to being linear and therefore training more quickly. This approach was described by Romero in [60], where the authors experiment with feeding extracted features to a linear SVM and other traditional models.

Another approach is to simply feed the output of the pre-trained model into a new layer with a classifier. In practice, this is done by creating a new model with the pre-trained model as a base and adding new fully-connected layers with a new classifier, as illustrated in figure 2.12. When doing this, it's very important to freeze the base model's layers to block any changes to their weights when training the new model.

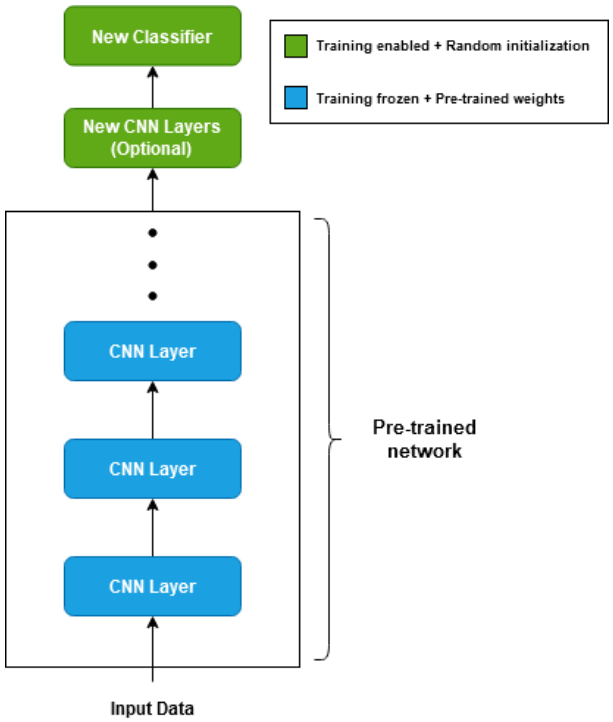


Figure 2.12: Illustration of a general example of a feature extraction approach to transfer learning.

2.3.2 Fine Tuning

Fine tuning is an approach to transfer learning in which the user continues training a pre-trained network on the new dataset it's being applied to.

In a process similar to feature extraction, a new model is created using the pre-trained network as a base, without the top. The user then adds a classifier for the new task as well as other new layers if necessary. When training the new model it's important to set a low training rate to fine tune to base model's parameters from their original values without losing important information previously learned [77].

Using appropriate hyper-parameters, such as low learning rate, this approach usually outperforms feature extraction [77]. However, it's also very easy to end up overfitting and losing previously learned knowledge. Sometimes, rather than retraining the whole model, part of the network is frozen before fine tuning as a way to prevent overfitting [77]. Usually the lower level layers are frozen as they tend to learn more general features than the higher level layers [11]. Figure 2.13 illustrates an example of a model with all but the last layer of the pre-trained CNN being frozen, in preparation for fine tuning.

2.3.3 Training and Optimization

When training a deep learning model, optimization is the process of finding the best set of weights in order to minimize the loss function.

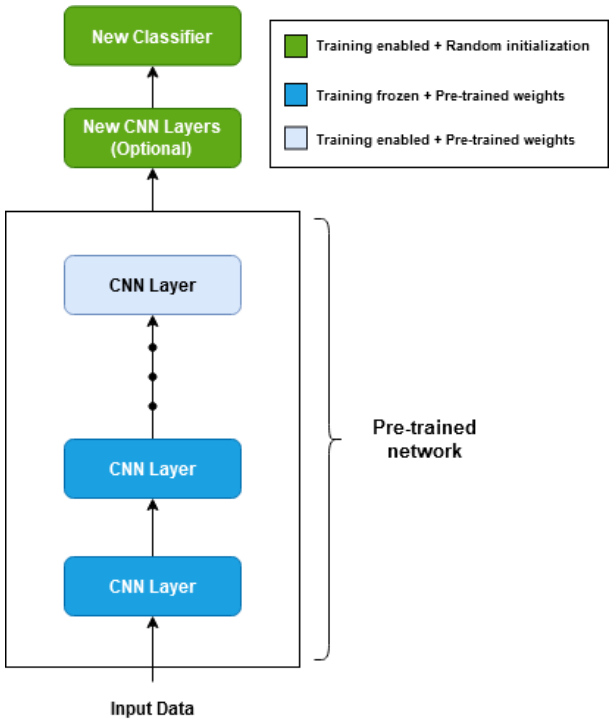


Figure 2.13: Illustration of a general example of a fine-tuning approach to transfer learning.

The gradient descent is the basic optimization algorithm and works by following the gradient of the loss function. The gradient of the loss function is a vector containing the partial derivatives for each dimension in the input space. Following this gradient leads in the direction of the best set of parameters for a given problem. One of the main problems with gradient descent happens when a loss function is non-convex and following the gradient ends up converging to a local minimum.

The stochastic gradient descent (SGD) provides a solution to that problem and increases the efficiency of the overall problem, but currently Adam is widely regarded as the best optimization algorithm for most classification problems. Adam [43], short for adaptive moments, is an extension to stochastic gradient descent and is gaining popularity over the years.

The authors describe Adam as a combination of two other extensions of SGD:

- **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients - especially useful in computer vision.

- **Root Mean Square Propagation** (RMSProp) that also maintains per-parameter learning rates that are adapted based on the decaying average of recent magnitudes of the gradients for the weight. This means that the algorithm does well on online and non-stationary problems.

The end result is an optimizer with decaying momentum that is computationally efficient and gives good results very quickly.

After each batch of training data goes through the network, Adam updates parameters θ using equation 2.6, which first requires the computation of equations 2.7, 2.8, 2.9, 2.10, and where η is the learning rate, β_1 and β_2 are exponential decay rates (or first and second momentum terms) for moment estimates and are respectively initialized at 0.9 and 0.999, ε is just a small value to prevent division by 0 (usually 10e-8), g is the gradient, t is a time-step initialized at 0 and increased by 1 after each batch, m and v are the first and second moment vectors.

$$\theta_{t+1} = \theta_t - \frac{\eta * \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.6}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.7}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.8}$$

$$m_t = (1 - \beta_1) g_t + \beta_1 m_{t-1} \tag{2.9}$$

$$v_t = (1 - \beta_2) g_t^2 + \beta_2 v_{t-1} \tag{2.10}$$

Gradients are calculated by computing the derivatives in each layer. The softmax function takes a vector as input and produces a vector as output, which means its' derivative can't just be directly computed. Instead, the partial derivative of a specific output j in respect to a specific input i is calculated:

$$\frac{\partial \hat{y}}{\partial x_j} = \begin{cases} \hat{y}_i(1 - \hat{y}_j), & \text{if } i = j \\ -\hat{y}_j . \hat{y}_i, & \text{if } i \neq j \end{cases}$$

Using back-propagation, the derivative of the cross entropy loss function in respect to a specific output after the softmax layer can be computed by multiplying the derivative of the loss function in relation to its direct output with the derivative of the softmax function in relation to the final output. Equation 2.11 shows the derivative of the cross entropy loss in relation to its own output i, equation 2.12 shows the multiplication of the loss function's derivative with the previously calculated softmax derivative.

$$\frac{\partial L}{\partial \hat{y}_i} = -\sum_i y_i \frac{1}{\hat{y}_i} \tag{2.11}$$

$$\begin{aligned}
\frac{\partial L}{\partial x_j} &= -\sum_{i \neq j} y_i \frac{1}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial x_j} - y_j \frac{1}{\hat{y}_j} \frac{\partial y_j}{\partial x_j} \\
&= -\sum_{i \neq j} y_i \frac{1}{\hat{y}_i} (-\hat{y}_i . \hat{y}_j) - y_j \frac{1}{\hat{y}_j} (1 - \hat{y}_j) \\
&= \sum_{i \neq j} y_i \hat{y}_j + y_j \hat{y}_j - y_j \\
&= \sum_{i} y_i \hat{y}_j - y_i \\
&\text{Since y is a one hot encoded vector, } \sum_i y_i = 1 \text{ resulting in:} \\
&= \hat{y}_j - y_i
\end{aligned} \tag{2.12}$$

This process is repeated for every layer all the way back to the start of the network, with each layer's derivative being calculated by multiplying with the previously calculated derivative.

2.4 Performance Metrics

Performance metrics are an effective way to evaluate the performance of machine learning algorithms.

One of the key concepts in classification performance is the confusion matrix (or error matrix), which is a way to visualize the machine learning model prediction versus the ground-truth labels.

Table 2.1 represents this concept adapted to the problem of this work. The predicted class indicates if the machine learning algorithm classified something as melanoma or nevi and the actual class represents the ground-truth data.

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	True Positive	False Negative
	Nevi	False Positive	True Negative

Table 2.1: Confusion Matrix

Analysing table 2.1 it can be said:

- **True Positive** (TP) is the number of times the algorithm correctly classified something as positive.
- **False Positive** (FP) is the number of times the algorithm incorrectly classified something as positive.
- **True Negative** (TN) is the number of times the algorithm correctly classified something as negative.

- **False Negative** (FN) is the number of times the algorithm incorrectly classified something as negative.
- The sum of TP, FP, TN and FN (T) is equal to the total number of classifications done by an algorithm.

With this data multiple metrics can be obtained:

- **Sensitivity** (SE), **recall, true positive rate** (TPR) or **probability of detection**.

$$SE = \frac{TP}{TP+FN} \tag{2.13}$$

- **Specificity** (SP) or **true negative rate** (TNR).

$$SP = \frac{TN}{TN+FP} \tag{2.14}$$

- **Precision** (Pre) or **positive predictive value**.

$$Pre = \frac{TP}{TP+FP} \tag{2.15}$$

- **Negative predictive value** (NPV).

$$NPV = \frac{TN}{TN+FN} \tag{2.16}$$

- **Miss rate** or **false negative rate** (FNR).

$$FNR = \frac{FN}{FN+TP} = 1-SE \tag{2.17}$$

- **Fall-out, false positive rate** (FPR) or **false alarm probability**.

$$FPR = \frac{FP}{TN+FP} = 1-SP \tag{2.18}$$

- **False discovery rate** (FDR).

$$FDR = \frac{FP}{FP+TP} = 1-Pre \tag{2.19}$$

- **False omission rate** (FOR).

$$FOR = \frac{FN}{FN+TN} = 1-NPV \tag{2.20}$$

- **Accuracy** (Acc).

$$Acc = \frac{TP+TN}{T} \tag{2.21}$$

- F_1 **score**. The harmonic mean of precision and sensitivity.

$$F_1 = 2 \times \frac{Pre \times SE}{Pre + SE} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.22}$$

- **Area Under the ROC Curve** (AUC)

  An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters, TPR and FPR, at different classification thresholds [22]. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

  The AUC measures the entire two-dimensional area underneath the entire ROC curve, providing an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example [22].

Both **sensitivity** 2.13 and **specificity** 2.14 are widely used in medicine and will be the most relevant when evaluating the results of the classification algorithms developed. In this context, sensitivity is the extent to which actual positives are not overlooked (low number of false negatives), and specificity is the extent to which actual negatives are classified as such (low number of false positives). The **AUC** is widely used in machine learning as an easy way to tell how much a model is capable of distinguishing classes, the higher the AUC the better. It will also be a prominent evaluation metric in this work.

Further, Brinker in [8] describes the creation of the first public melanoma classification benchmark for both non-dermoscopic and dermoscopic images for comparing artificial intelligence algorithms with diagnostic performance of 145 or 157 german dermatologists. This benchmark will be used to compare the results obtained in this work.

2.5 Datasets and related work

The HAM10000 ("Human Against Machine with 10000 training images") dataset was created to assist in the training of neural networks for automated diagnosis of pigmented skin lesions. The dataset consists of 10015 dermatoscopic images which are publicly available through the ISIC archive. Specifically, it consists of 327 images of actinic keratoses, 514 basal cell carcinomas, 1099 benign keratosis, 115 dermatofibroma, 1113 melanomas, 6705 melanocytic nevi and 142 vascular skin lesions [75]. In comparison, the PH2 dataset consists of 40 melanomas and 160 melanocytic nevi [49].

The HAM10000 is the biggest dataset of skin lesions publicly available while also boasting reliable ground-truth (more than 50% of lesions have been confirmed by pathology, while the ground truth for the rest of the cases was either follow-up, expert consensus, or confirmation by in-vivo confocal microscopy [75]), and as such it will be the dataset used in this dissertation.

As seen in table 2.2, results from classification algorithms described in other studies vary widely and, due to the use of different metrics, can be hard to directly compare. Looking at the PH2 dataset, there are multiple traditional machine learning classification algorithms using hand-crafted methods for feature extraction and boasting good results. However, with the larger HAM10000 dataset it can be seen that deep learning is much more commonly used. This is particularly noticeable when looking at the ISIC challenge 2018 leaderboards for "Task 3: Lesion Diagnosis", available in [9]. The approaches used in this challenge are predominantly related to deep learning, including the ones with the best results.

Due to the results of the ISIC challenge favoring deep learning approaches and to the high level of domain expertise and time spent needed to create an hand-crafted method for feature extraction that is good enough to obtain the best results seen in 2.2, this dissertation will focus on deep learning approaches to classify melanomas.

Several studies have been published confirming that AI can perform just as well, or even better, than trained professionals when it comes to the detection health conditions, including skin cancers.

In the paper ***Deep Learning and Handcrafted Method Fusion: Higher Diagnostic Accuracy for Melanoma Dermoscopy Images*** [26], a complex conventional image processing based classifier, which detects various visible lesion features and uses other pathological and patient information, yields an AUC of 0.90 while a simpler ResNet-50 deep learning based classifier alone yields an AUC of 0.83. Fusing both methods, an AUC of 0.94 was achieved. For the deep learning approach, Hagerty took the feature map with 2048 elements that ResNet50 outputs before the FC layers and used principal component transform to select the first 1024 principal components, which were then ran through a regression learning algorithm, yielding an AUC of 0.83. The pretrained deep learning model is only being used for feature extraction, which poses a limitation to the performance since the images used are fairly different from those that the model was original trained on. Fine tuning the weights of the pretrained network should yield higher performance.

Esteva et al., in ***Dermatologist-level classification of skin cancer with deep neural networks*** [21], train a CNN using 129450 clinical images from 2032 diseases achieving performance on par with 21 tested board-certified dermatologists. The model developed is based on the Google Inception v3 CNN, pretrained on the ImageNet dataset and fine-tuning strategies are used as transfer learning techniques. Testing the performance with 1010 dermoscopic melanoma images, their algorithm obtained an AUC of 0.94.

Codella et al., in ***Deep Learning, Sparse Coding, and SVM for Melanoma Recognition in Dermoscopy Images*** [13] compares CNN and sparse coding against a SOTA ensemble modeling of low-level visual features, finding results that slightly favour transfer learning with CNNs over sparse coding but an even higher effectiveness increase when fusing all models. This study uses an earlier version of the HAM10000 dataset with 2624 total images, of which 334 are melanomas, 144 atypical nevi and 2146 images of benign lesions. The classification problems here are classifying images as malignant (melanomas) or benign (atypical nevi and benign skin lesions) and as malignant or atypical nevi. The results shown here are from the second problem, since it is the more relatable to the problem addressed in this work. The study used a pretrained Caffe CNN to

extract features that were then used to train a linear SVM, yielding a performance of 72.5% accuracy, 72.5% sensitivy and 72.3% specificity. Performance is heavily limited by the dataset and CNN used, which may have been good options at the time but currently there are better options. Additionally, fine tuning the model rather than extracting features to a linear SVM should improve the performance.

Brinket et al. published two separate studies related to melanoma and melanocytic nevi detection in [7, 6]. Both studies used pre-trained ResNet50 models and were retrained in the same way with varying learning rates to fine tune the weights on every layer. The main difference between the two studies lies in the dataset. In [7] all melanoma and melanocytic nevi images (a total of 12378 images) are used, meaning the training is done with an unbalanced dataset, whereas in [6] some nevi images were removed in order to balance the dataset, ending with 4204 total images. Both studies used the stochastic gradient descent optimization algorithm, possibly giving worse performance than they would if using the Adam optimization algorithm.

Dataset	Dataset Preparation	Method for Feature Extraction	Classifier	SE	SP	Pre	ACC	AUC	Ref.
Private		Deep Learning (Inception V3)	Softmax					0.94	[21]
HAM10000 [14, 75]	Data augmentation, Additional training images	Deep Learning (ResNet50) + Hand-Crafted	Softmax					0.94	[26]
	Exclusion of some images	Deep Learning (ResNet50)	Softmax	89.4%	64.4%				[7]
	Exclusion of some images	Deep Learning (ResNet50)	Softmax	82.3%	77.9%				[6]
		Deep Learning (ResNet50)	gcforest			80.4%			[59]
		Deep Learning (Caffe CNN)	SVM	72.5%	72.3%		72.5%		[13]
PH2 [49]		Deep Learning (Vgg19)	Logistic Regression			85.71%	92.5%		[47]
		Hand-crafted + Dictionary (BoF)	Random Forests	98.0%	90.0%				[3]
	Data augmentation, Exclusion of some images	Hand-crafted	kNN	96.0%	83.0%				[63]
		Hand-crafted	SVM	96.0%	97.0%				[66]
	Additional training images	Hand-crafted	Softmax	83.3%	95.0%				[76]

Table 2.2: Melanoma classification scores using two publicly available datasets. Compiled information on PH2 dataset obtained from [4].

Chapter 3

Methodology

3.1 Setup

This project was developed using a laptop of the Acer Predator line, with an Intel i7 8th generation CPU, CUDA enabled NVIDIA GeForce GTX 1060 GPU, 16 GB of RAM and Windows 10 operating system.

In an anaconda environment for python, the latest versions of the following packages were used: Keras [12], Tensorflow [1], Scikit-learn [56], NumPy [53] and Matplotlib [34].

Keras is a high level neural networks API built on top of Tensorflow 2.0, "covering every step of the machine learning workflow, from data management to hyperparameter training to deployment solutions" [12]. It's particularly useful for starting out with deep learning due to being user-friendly, having quality APIs and some built-in pre-trained models. **Tensorflow** is an "end-to-end open source platform for machine learning" [1]. It has an extensive amount of tools and libraries that lets researchers and developers easily build and deploy machine learning systems. **Scikit-learn** is an open source python library with Simple and efficient tools for predictive data analysis. It can be used in the context of classification, regression, model selection, pre-processing, etc. **NumPy** is a scientific package for Python and in machine learning two of its' most important features are dealing with array objects and linear algebra routines. **Matplotlib** is a library focused on creating static, animated, and interactive visualizations in Python.

3.2 Dataset preparation

The dataset chosen for this project is the HAM10000, the dataset created for the ISIC Challenge 2018 [14, 75]. The original dataset contains a total of 10015 images of 7 different categories in a single folder, with ground-truth data provided in a separate csv file.

As this project is only concerned with classifying melanoma and nevi lesions, the dataset was prepared using python scripts.

First, all images that belong to any category other than *melanoma* or *nevi* were deleted, leaving us with a total of 7818 images. These were then separated into separate folders for each category

and then further separated, randomly, into training, validation and testing folders, with each getting, respectively, 60%, 20% and 20% of the images.

The final layout of the dataset is shown in table 3.1.

	Training	Validation	Testing
Melanoma	669	222	222
Nevi	3729	1488	1488

Table 3.1: Dataset layout

3.3 Data loading and augmentation

Using the *ImageDataGenerator* class provided by Keras, data is loaded in small batches into memory using the *flow_from_directory* method, which is prepared for the layout that the dataset was divided in. This allows the training of datasets of any size while using little physical memory. Another advantage of using this class is the possibility of random and automatic generation of augmented versions of images.

Images are loaded and sent as input to the deep learning model in batches of 32, a number chosen after some testing due to it's performance in the generalisation of the model and low physical memory usage. Before being used as input, all images are resized to 224x224, the input size for both the VGG16 and ResNet50 models, and maintain the color mode as rgb. Class mode set to categorical. Since there are only two classes it could be binary, but categorical is the default and can be easily expanded to more classes. The only difference is the format of the output.

Training and validation images are loaded in a random order to improve generalization. Testing images are **not** loaded randomly so they can be matched to labels after evaluation. In addition, in order to improve generalisation of the model, training images undergo a random data augmentation process to increase the diversity of data used without collecting new data. The data augmentation techniques used were chosen while attempting to not destroy the important information in the images, for example, cropping images is not used to avoid cropping the borders of lesions, which are an important feature. The data augmentation is done in real-time and works by taking an original batch of images and replacing it with a newly transformed batch, meaning no new images are added directly into each epoch (a full run of the entire dataset). Each epoch then uses the same number of images that are transformed in different ways each time, giving an undefined total of different images used. For each image, data augmentation techniques are randomly chosen from the following:

- Rotation of up to 15 degrees (rotation amount doesn't seem to cause any problems in training and could be different).
- Random width shift with range of 10% of total width (shifts are set to a max of 10% to avoid losing important information).

- Random height shift with range of 10% of total height.
- Random horizontal flips.
- Random vertical flips.

3.4 Transfer Learning

Both the VGG16 and ResNet50 architectures were considered for this project. During initial tests for this project, models based on the ResNet50 were not converging and due to lack of time to make it work, combined with the VGG16 having a simpler architecture that showed a stable generalisation capacity, led to choosing its feature extraction layers (as shown in figure 3.1, as the base model to use, in detriment of the ResNet50.

Both the feature extraction and fine tuning approaches to transfer learning were tested and exact configurations are described below.

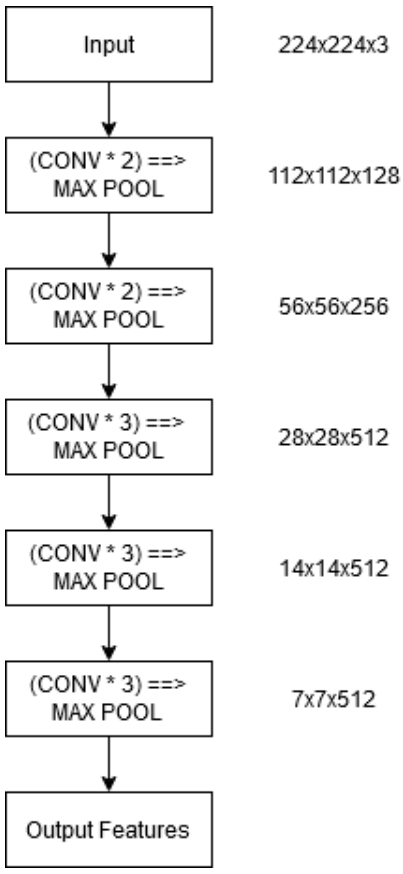


Figure 3.1: Compact visualization of the VGG16 model without top classification layers.

3.4.1 Feature Extraction

In order to prepare the model to the feature extraction approach to transfer learning, training in all layers of the base model is frozen and new layers are added. For the addition of classification layers, two different alternatives were explored:

Following the traditional method, a flatten layer is added, then two fully connected layers, followed by a third fully connected layer with 2 nodes and a softmax activation function. The number of nodes in the first two layers is variable and influences the efficacy of the model. The node amounts of 512, 1024 and 2048 will be tested to find the best value. After each of the first two FC layers, a dropout layer is also added to help reduce overfitting by randomly setting around 20% of input units to 0, followed by a batch normalization layer. Following the newer method, a global average pooling layer was added, followed by a softmax activated fully connected layer with 2 nodes. Both methods are illustrated in figure 3.2, with the left having FC * 2 to represent both fully connected layers and hidden dropout and batch normalization layers.

3.4.1.1 Training process

The training process is done by inputting training images in batches of 32. After each batch has finished traversing the entire network, the cross entropy loss is calculated and back-propagated, updating the weights in each of the fully connected layers. At the end of each epoch the model's performance and loss is evaluated using validation data. This process is repeated for up to 20 epochs, with weights being updated using the Adam optimizer with a learning rate of 0.0001, chosen for providing good results after initial testing with several different learning rates, and a β_1 of 0.9, $\beta 2$ of 0.999 and ε of 10e-8, as suggested for image classification in Adam's introduction paper [43]. The metric used to follow the learning progress is AUC. Although it does not provide the same results during training as it does when calculated after evaluation, it is still preferred in this work due to the traditional accuracy not accounting for the unbalanced dataset causing it to be often stuck at around 85%.

Callbacks are used to:

1. Reduce the learning rate to 30% of its previous value if loss doesn't improve in 3 consecutive epochs.
2. Save the weights of the model after each epoch so it can be recovered if the model starts to overfit.
3. Stop training early if validation loss doesn't improve in 8 consecutive epochs.

Some of the most important hyper-parameters are the class weights. Since the dataset is heavily unbalanced, with default class weights the model tends to always classify images as Nevi as that's the easiest way to get low loss. The class weights shown in table 3.2 are balanced using an "heuristic" inspired by King in [42]:

$$weight = \frac{n_{samples}}{n_{classes} * n_{samples\_with\_class}} \tag{3.1}$$

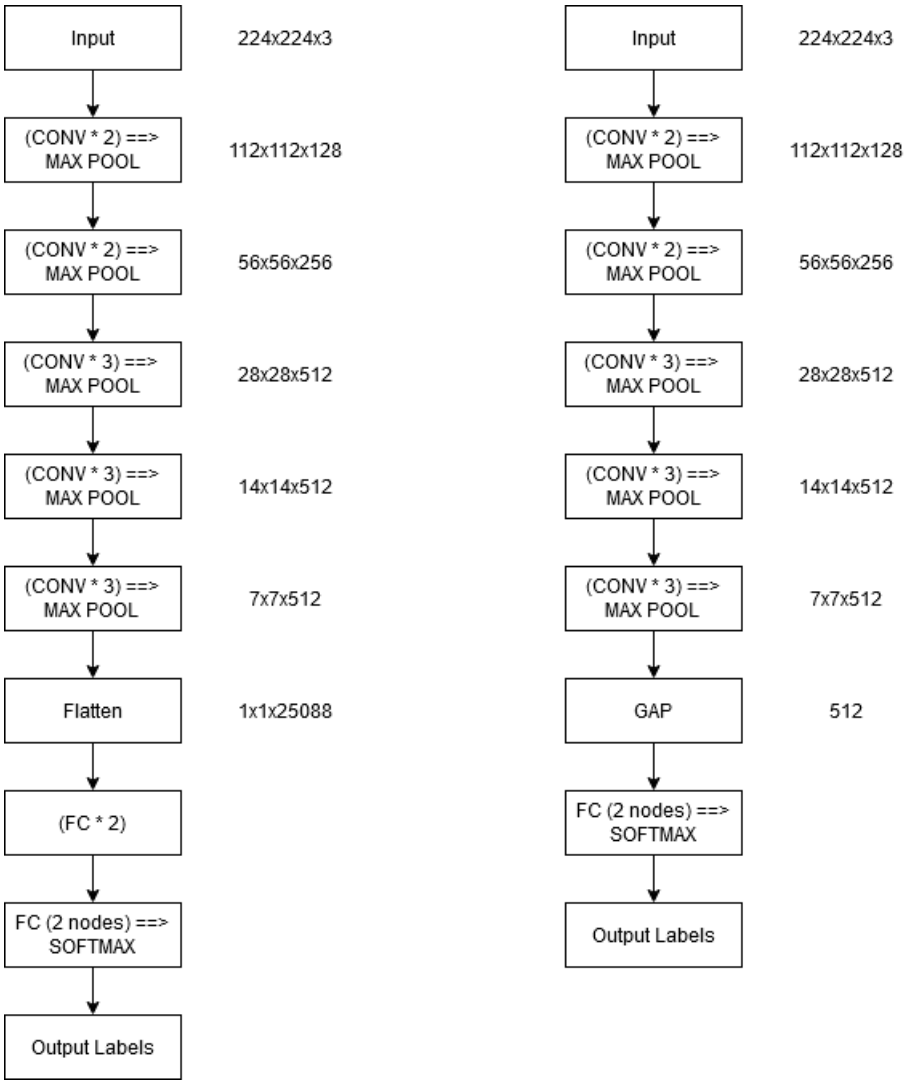


Figure 3.2: Compact visualization of the new model with new top classification layers added with traditional method (left) and newer method (right).

	Melanoma	Nevi
Class Weight	3.287	0.589

Table 3.2: Estimated class weights

3.4.2 Fine Tuning

Rather than using a model with newly added classification layers with randomized weights, this training consists in fine tuning the model with fully connected layers that showed the best results in the feature extraction method.

The reasoning for this choice is simple: since the classification layers are already trained, a fine tuning approach should be able to further increase the efficacy of the model.

Additionally, a second test was done using a second model created following the newer method used by the ResNet50: a global averaging pool layer is added to the base model, followed by a fully connected layer with 2 nodes and a softmax activation function. This model didn't go through the feature extraction process since it adds very few trainable parameters and performed very poorly without fine-tuning.

Regarding unfreezing, two different tests will be done separately: one with training with a fully unfrozen base model and another with only the last block of 3 convolutional layers unfrozen.

3.4.2.1 Training process

The differences in this training process in relation to the one in feature extraction is the change in number of epochs to up to 10, learning rate to 0.00005 and the removal of the callbacks that reduce the learning rate and stop training early. Additionally, the weights of the final classification layers are not the only ones being updated, they will also be updated on the unfrozen layers.

3.5 Changing weights

While the weights calculated using the heuristic described earlier already balance the dataset, one could argue that detecting melanoma is more important than detecting nevi. After manually setting new class weights shown in table 3.3 to further increase the importance of melanoma and decrease nevi, the best performing test was rerun to see if there's any significant improvement.

	Melanoma	Nevi
Class Weight	3.8	0.55

Table 3.3: Manually changed class weights

In summary, several tests will be done to evaluate performance of different models:

1. Training models with frozen feature extraction layers and trainable fully connected classification layers with 512, 1024 and 2048 nodes each (3 tests).

2. Fine tuning the best performing model with the last 3 convolutional layers unfrozen, as well as with all layers unfrozen (2 tests).

3. Fine tuning a model with a global average pooling layers substituting the fully connected layers, with 3 convolutional layers unfrozen and with all layers unfrozen (2 tests).

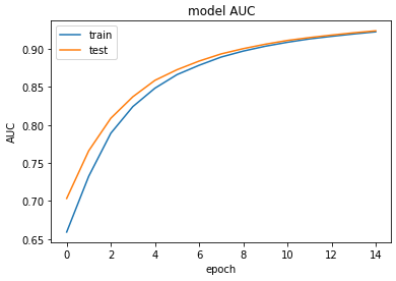
4. Manually setting class weights to increase importance of melanoma and repeating the training process used in the best performing model.

Chapter 4

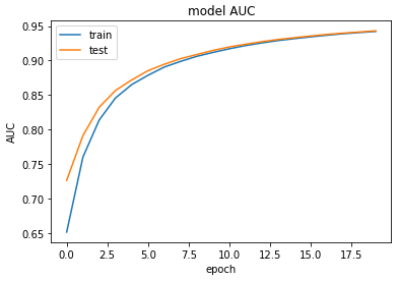
Results

4.1 Feature Extraction

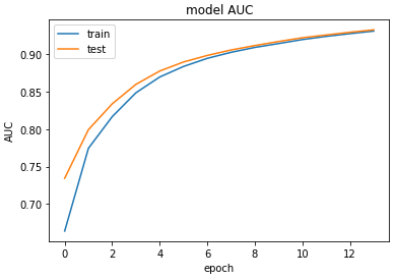
The models with 512, 1024 and 2048 nodes in the fully connected layers performed in a generally similar way as seen in figures 4.1, 4.2 and 4.3, with evaluation performance slightly increasing with the increase in nodes. It's important to mention that the model with 512 nodes in each FC layer achieved its best performance after 6 epochs and began slowly deteriorating after that point, the model with 1024 nodes achieved its best performance after the last epoch and the model with 2048 nodes after 12 epochs. The evaluation of each model is done using a snapshot of the model at its highest performance and is shown below in figure 4.3 and tables 4.1, 4.2 and 4.3.



(a) FC layers with 512 nodes



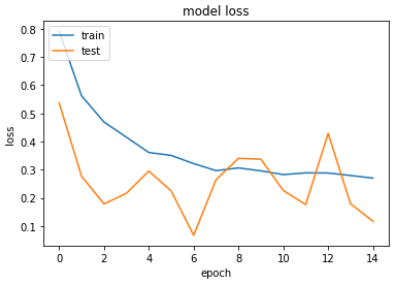
(b) FC layers with 1024 nodes



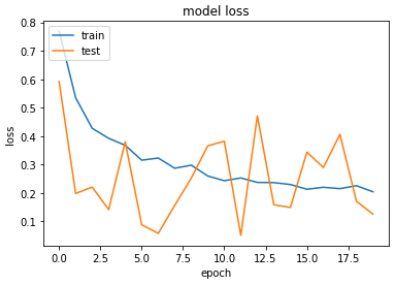
(c) FC layers with 2048 nodes

Figure 4.1: Plots of validation AUC in training of models with different FC layers

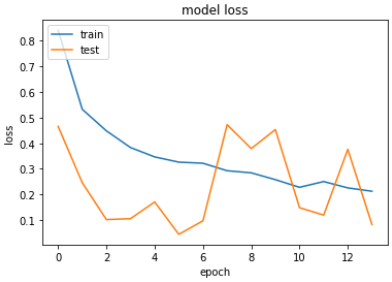
Analysing the confusion matrices 4.1, 4.2 and 4.3, it can be said:



(a) FC layers with 512 nodes

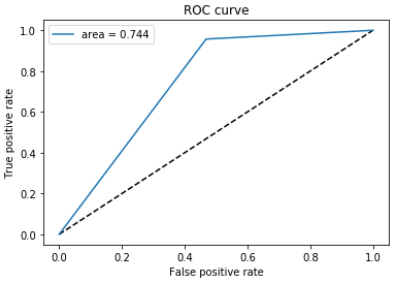


(b) FC layers with 1024 nodes

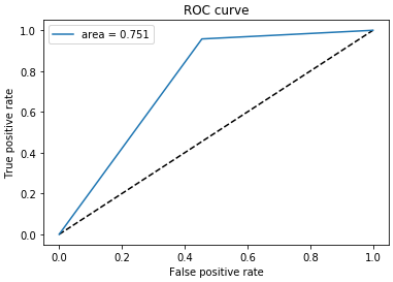


(c) FC layers with 2048 nodes

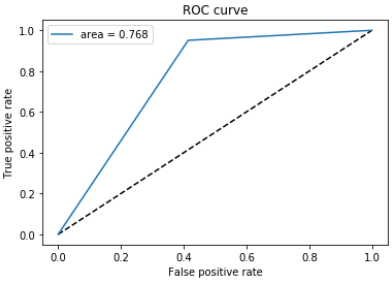
Figure 4.2: Plots of validation loss in training of models with different FC layers



(a) FC layers with 512 nodes



(b) FC layers with 1024 nodes



(c) FC layers with 2048 nodes

Figure 4.3: Plots of ROC curve in the evaluation of models with different FC layers

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	118	104
	Nevi	64	1424

Table 4.1: Confusion Matrix of model with FC layers with 512 nodes

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	121	101
	Nevi	63	1425

Table 4.2: Confusion Matrix of model with FC layers with 1024 nodes

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	130	92
	Nevi	73	1415

Table 4.3: Confusion Matrix of model with FC layers with 2048 nodes

- The model using 512 nodes has 53.15% sensitivity and 95.7% specificity
- The model using 1024 nodes has 54.5% sensitivity and 95.76% specificity
- The model using 2048 nodes has 58.55% sensitivity and 95.09% specificity.

Although the results suggest that further increase in the number of nodes of the fully connected layers can improve the results, it also makes training considerably more computationally expensive. For comparison, the model with 2 FC layers with 2048 nodes each has 55,590,914 trainable parameters and 14,722,880 non-trainable parameters, where the non-trainable parameters belong to the base model, while the model with 2 FC layers with 512 nodes each only adds 13,111,298 trainable parameters.

4.2 Fine tuning

4.2.1 Base model partially trainable

The model using a global average pooling layer for classification immediately shows great results when fine-tuning the last block of 3 convolutional layers of the base model. Towards the end of training it does begin to overfit but its best performance, achieved after 8 epochs, showed great results. Analysing table 4.4, it can be said that the model has 75.67% sensitivity and 90.72% specificity, which are the best results obtained in these tests until now.

The performance of the model with 2048 nodes fully connected layers has also shown great performance after partially fine tuning the base model. Analysing table 4.5, we can say this model now has 68% sensitivity and 94.76% specificity, a great improvement from the previous 58.55% and 95.09%.

The AUCs of both models are now pretty comparable, with a slight edge to the model using the GAP layer at 0.832 (seen in figure 4.8) versus the other model's 0.814 (seen in figure 4.9).

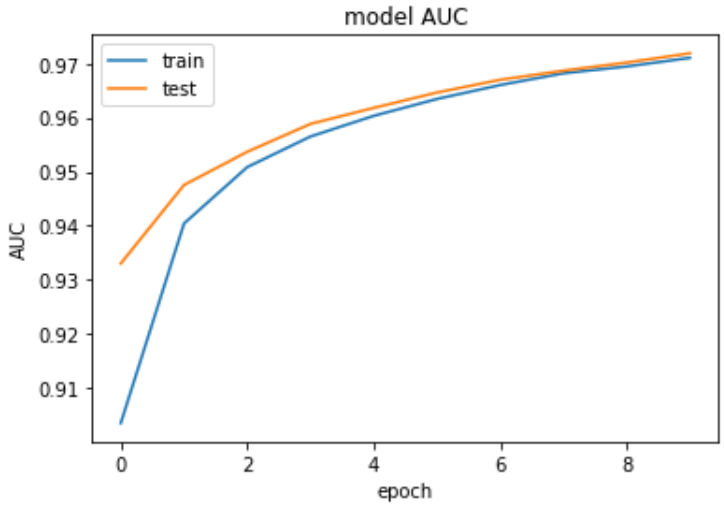


Figure 4.4: Plot of validation AUC in training of model with GAP layer and base model **partially** trainable.

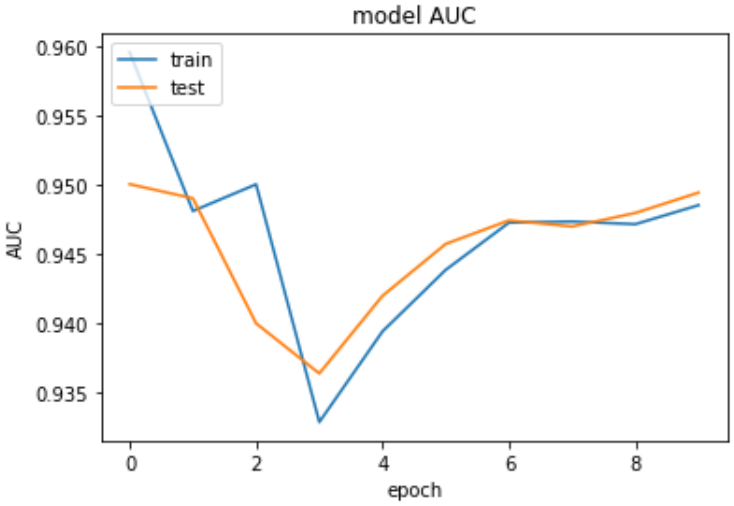


Figure 4.5: Plot of validation AUC in training of model with FC layers with 2048 nodes and base model **partially** trainable.

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	168	54
	Nevi	138	1350

Table 4.4: Confusion Matrix of model with GAP layer and base model **partially** trainable.

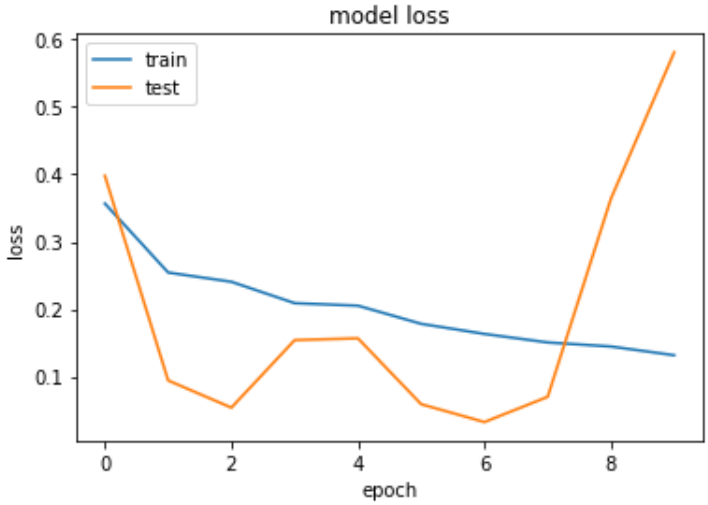


Figure 4.6: Plot of validation loss in training of model with GAP layer and base model **partially** trainable.

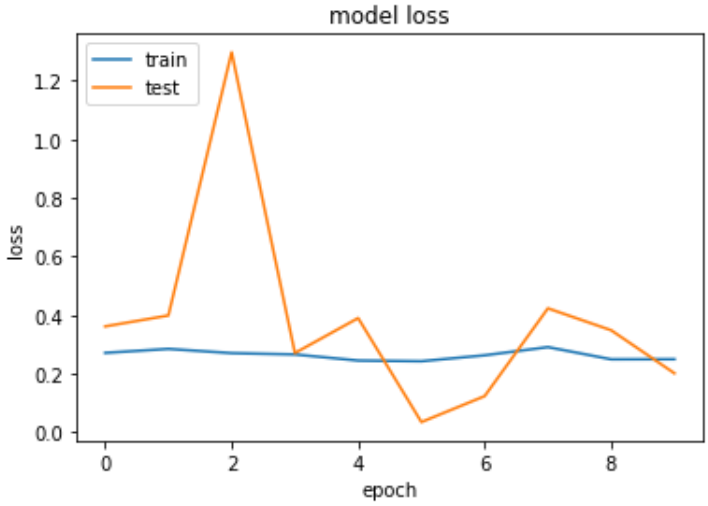


Figure 4.7: Plot of validation loss in training of model with FC layers with 2048 nodes and base model **partially** trainable.

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	151	71
	Nevi	78	1410

Table 4.5: Confusion Matrix of model with 2048 nodes FC layers and base model **partially** trainable.

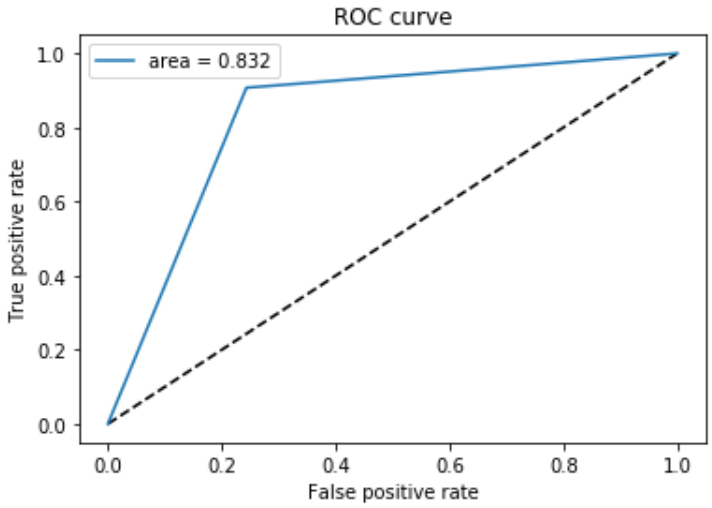


Figure 4.8: Plot of ROC curve in the evaluation of model with GAP layer and base model **partially** trainable.

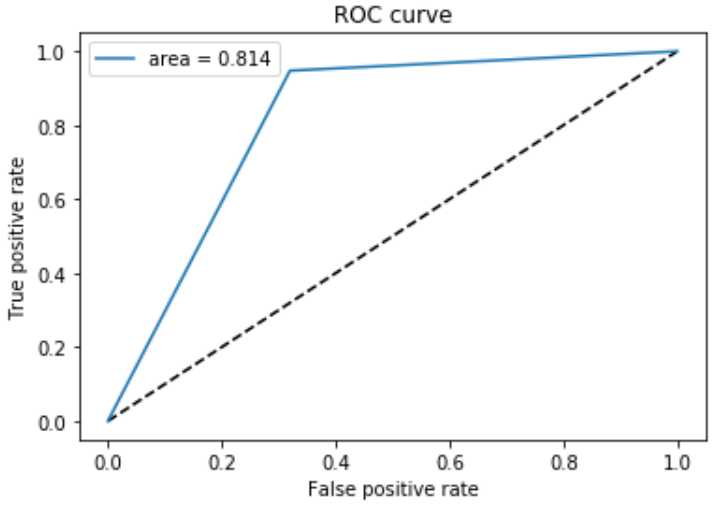


Figure 4.9: Plot of ROC curve in the evaluation of model with FC layers with 2048 nodes and base model **partially** trainable.

4.2.2 Base model fully trainable

With a fully trainable base model, using a GAP layer yields similar results to the previous test with a partially trainable base model. Like in the previous test, this model achieved its best performance after 8 epochs with 73.87% sensitivity, slightly worse than before, and 92.94% specificity, slightly better (calculated using the confusion matrix in tables 4.6). The AUCs of both GAP models are almost equal, with 0.832 on the previous test (shown in figure 4.8) and 0.834 now (shown in figure 4.14). The training on this model showed no drop in training AUC, seen in figure 4.10, and the loss seen in figure 4.12 starts to increase towards the end, possibly due to beginning to overfit.

Performance of the FC layer model with fine-tuning dropped heavily using a fully trainable base model vs a partially trainable base model. Performance shown in table 4.7 and figure 4.15 are taken from a snapshot saved after epoch 4, before the drop in validation AUC and spike in validation loss seen in figures 4.11 and 4.13, and is better than the performance registered after the full 10 epochs. This drop in performance may be due to complexity of the model being too high causing overfitting, with too many trainable parameters for the amount of features that can be detected with this dataset.

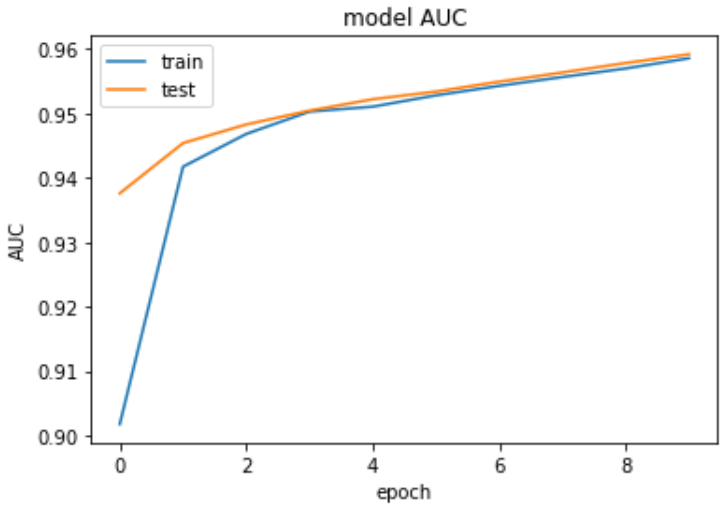


Figure 4.10: Plot of validation AUC in training of model with GAP layer and base model **fully** trainable.

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	164	58
	Nevi	105	1383

Table 4.6: Confusion Matrix of model with GAP layer and base model **fully** trainable.

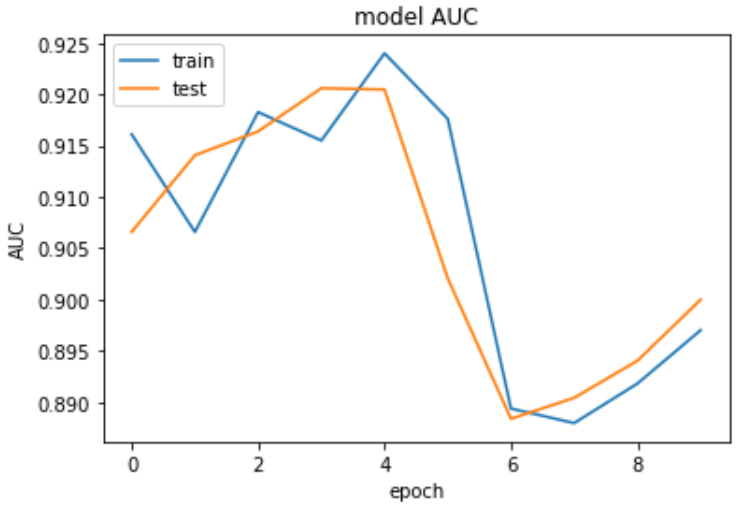


Figure 4.11: Plot of validation AUC in training of model with 2048 nodes FC layers and base model **fully** trainable.

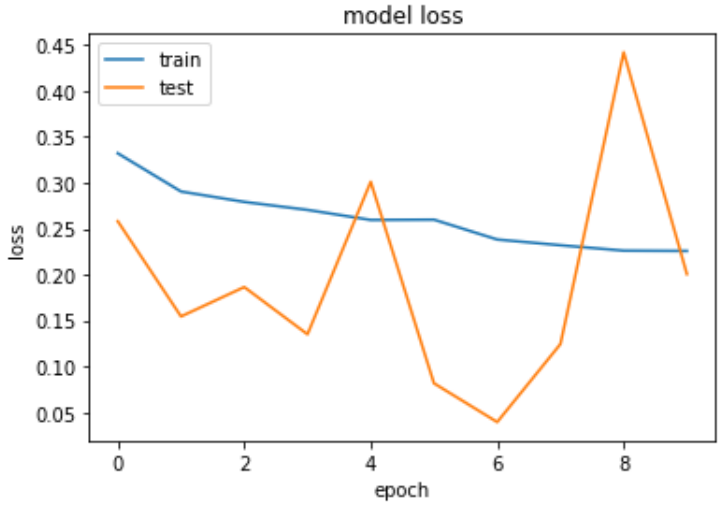


Figure 4.12: Plot of validation loss in training of model with GAP layer and base model **fully** trainable.

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	112	110
	Nevi	58	1430

Table 4.7: Confusion Matrix of model with 2048 nodes FC layers and base model **fully** trainable.

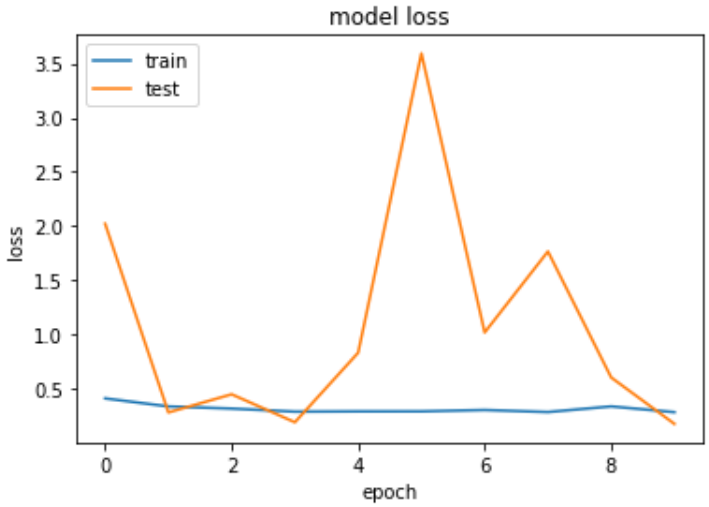


Figure 4.13: Plot of validation loss in training of model with 2048 nodes FC layers and base model **fully** trainable.

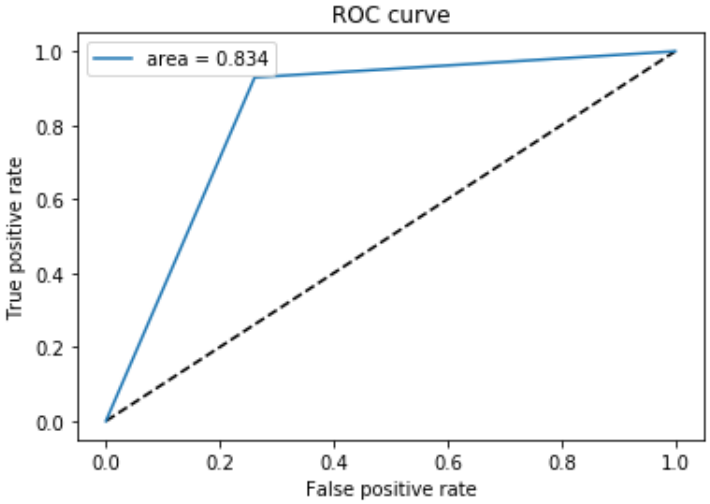


Figure 4.14: Plot of ROC curve in the evaluation of model with GAP layer and base model **fully** trainable.

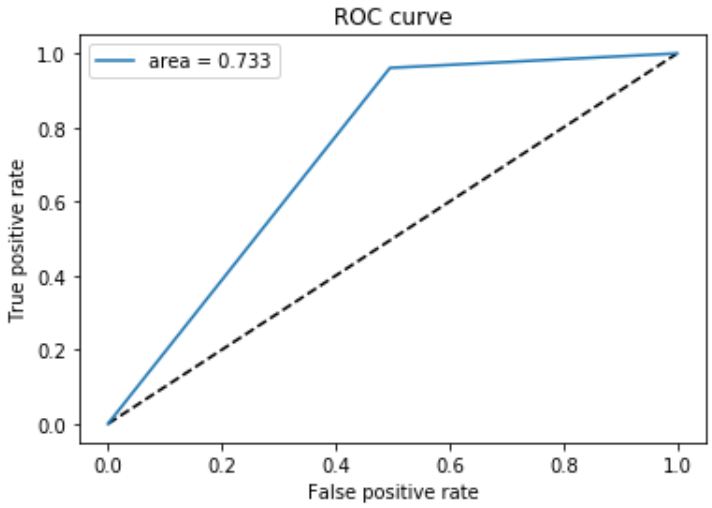


Figure 4.15: Plot of ROC curve in the evaluation of model with 2048 nodes FC layers and base model **fully** trainable.

4.3 Changing class weights

At this point the best performances have been achieved using models with a GAP layer. Fine-tuning with a partially or fully trainable base model yields similar results, with a slight edge to the fully trainable base model. However, these results still have considerably higher specificity than sensitivity. For a final test, the fully trainable model is rerun with a GAP layer with different class weights, giving heavier focus on correct melanoma detection.

Using a snapshot of the model after 4 epochs of training gives a nicely balanced performance shown in table 4.8 and figure 4.16. Analysing the confusion matrix, a sensitivity of 80.6% and specificity 86.42% can be calculated. After this point the model continues to give even higher focus on melanoma, ending on roughly 95% sensitivity and 72% specificity after the full 10 epochs, which is a little too unbalanced.

		Predicted Class	
		Melanoma	Nevi
Actual Class	Melanoma	179	43
	Nevi	202	1286

Table 4.8: Confusion Matrix of model with GAP layer and base model **fully** trainable, with manually set class weights.

In summary, using fully connected layers to handle classification does provide competitive results when carefully fine-tuning the model, as seen in figure 4.9 and table 4.5. The main disadvantages of this approach are the higher computational cost due to higher amount of parameters that need to be learned and the ease of messing up training with too much complexity as seen in figures 4.11, 4.13 and 4.15. Substituting the fully connected layers with a global average pooling

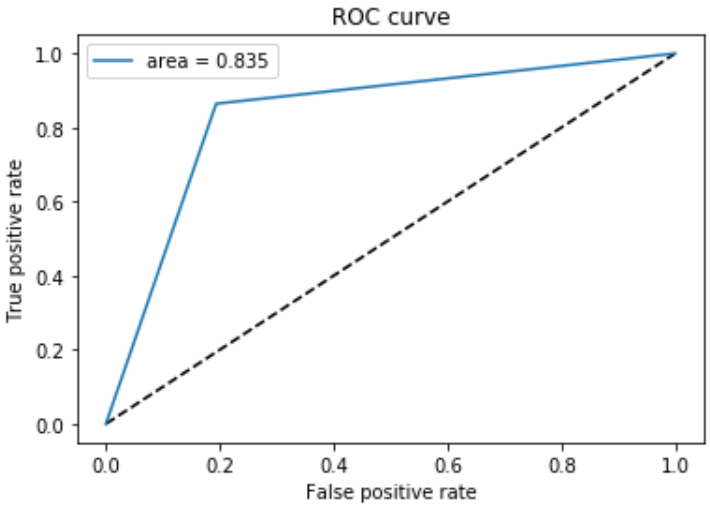


Figure 4.16: Plot of ROC curve in the evaluation of model with GAP layer and base model **fully** trainable, with manually set class weights.

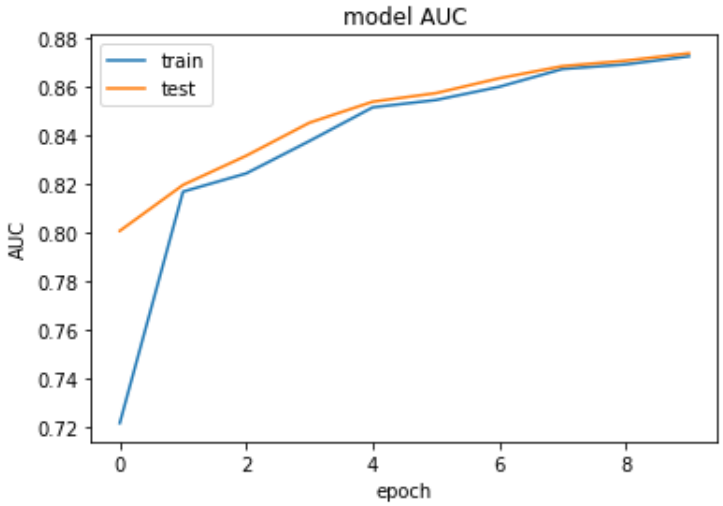


Figure 4.17: Plot of validation AUC in training of model with GAP layer and base model **fully** trainable, with manually set class weights.

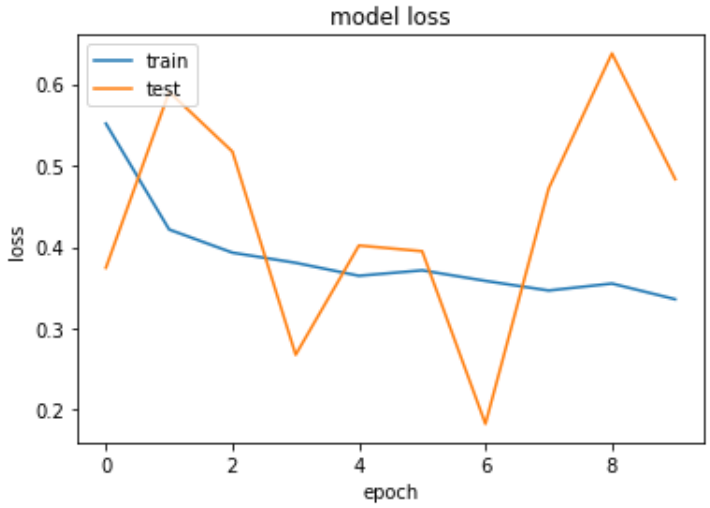


Figure 4.18: Plot of validation loss in training of model with GAP layer and base model **fully** trainable, with manually set class weights.

layer not only reduces the computational cost significantly but even ended up performing slightly better with both partial and fully trainable models, as seen in figures 4.8 and 4.14. Additionally, changing class weights to more heavily punish wrong classifications in melanomas showed a significant sensitivity increase, with an expected and not too high loss in specificity. Since the AUC seen in figure 4.16 is similar to the one seen in figure 4.14, it can be concluded that the loss in specificity was compensated by the increase in sensitivity.

Chapter 5

Conclusions and future work

This dissertation focused on deep transfer learning approaches for melanoma detection, with some interest in assessing how certain hyper-parameter changes and fine tuning approaches affected the performance of the deep learning models.

An initial introduction to melanoma and skin cancer in general showed that melanoma is one of three main types of skin cancer, which also include basal cell carcinomas and squamous cell carcinomas, and shows similar features to the benign skin lesion melanocytic nevus. Distinction between melanomas and melanocytic nevi can be particularly difficult as they can have similar features and nevi can even later evolve into melanomas, making this distinction a very important objective. In reviews of similar works it was concluded that, to our knowledge, classification of other skin lesions present in the HAM10000 dataset is more easily done than classification of melanoma. This can be seen in the ISIC2018 challenge leaderboards where even the best performing model had considerably lower rate of detection of melanomas than other lesions, at 77.8% sensitivity for melanomas vs 85% or higher for the other lesions [9]. These results led to the conclusion that distinction between melanomas and melanocytic nevi need further work, leading this work to focus on the classification of these two lesions rather than all seven lesions present in the HAM10000 dataset. This decision is also supported by the fact that this dataset contains a much smaller number of images of other lesions, which makes it even more unbalanced.

A review of the state-of-the-art in deep learning and transfer learning approaches was done, detailing some of its advances since the origin of the term deep learning. While deep learning requires vast amounts of data for training, using pre-trained weights in a transfer learning approach allows the use of convolutional neural networks in problems with much smaller datasets. This review shows that, to our knowledge, there are three main ways to handle classification after the convolutional layers detect features. The traditional way is to use fully connected layers to obtain combinations of the features detected previously and works reliably, despite being computationally expensive. A global average pooling layer can be used instead of fully connected layers and works by taking the average of each feature map from the output of the last convolutional layer and feeding the resulting vector directly into a softmax layer for classification. Doing this requires the amount of feature maps to be the exact same as the amount of classes to detect, which is rarely

an option in transfer learning. The third approach consists of using a global average pooling layer and feeding the resulting vector into a softmax activated fully connected layer with as many nodes as there are classes to detect.

A review of similar works showed that there are already various approaches to the classification of melanomas and melanocytic nevi. However, comparison between different works is often difficult due to mainly two issues:

1. The use of different types of metrics don't allow a direct comparison of results.

2. Even if metrics are the same, many different details in a model, such as learning rates and different combinations of classification layers, can cause the performance to vary wildly so it becomes difficult to ascertain the reason for a difference in performance between two different works.

One thing in common with most works is the use of transfer learning in one way or another to obtain competitive results. Traditional machine learning can perform just as well and even slightly better than deep learning, as shown by Hagerty in [26], but these require very extensive work in the feature extraction phase, with the assistance of dermatology professionals.

The aim of this dissertation was not only to develop a deep learning model with competitive results, but also to assess how different changes affect the performance. As such, different tests were run to conclude:

1. Using fully connected layers in the classification part of a network, the results suggest that, between 512 and 2048, it is best to have a higher amount of nodes.

2. When fine tuning a model with a high amount of parameters in the fully connected layers, the results suggest that it is best to unfreeze only the last few layers of the base model, rather than its entirety.

3. As expected, models using a global average pooling layer instead of fully connected layers yielded better and faster performance, both when fine tuning a partially unfrozen model and a fully unfrozen model.

4. Changing class weights from balancing the dataset to slightly favoring the underrepresented class can yield better results but will also quickly completely unbalance results.

The best performances obtained, with changed class weights, 80.6% sensitivity, 86.42% specificity and 83.5% AUC after 4 epochs of training and 95% sensitivity, 72% specificity and 83.7% AUC after 10 epochs. For comparison, the benchmark described in [8] shows 157 dermatologists obtaining sensitivity of 74.1%, specificity of 60.0% and an AUC of 0.67 using dermatoscopic images (the same type used in the HAM10000 dataset) and 145 dermatologists with a sensitivity of 89.4%, specificity of 64.4% and AUC of 76.9% using non-dermoscopic images. After just 4 epochs of training the model developed in this dissertation only has lower sensitivity than the benchmark and after 10 epochs all three metrics are higher. While using balanced weights, the

best performance has 73.87% sensitivity, 92.94% specificity and 83.4% AUC. Although the sensitivity is lower than in the benchmark, the AUC and specificity are still higher. Comparing to related works, the results obtained are not surprising but show promise. Hagerty in [26] obtained an AUC of 83% using deep transfer learning alone, but improved it to 0.94 by fusing both deep and traditional machine learning methods. Brinker in [6] obtained a sensitivity of 82.3% and 77.9% specificity which is similar to the best performance obtained in this work.

Future work of this dissertation could include the addition of further parameter testing. Further tuning of hyper-parameters can possibly help improve the developed model to yield even better results. Specifically, the learning rate has a major effect in learning and was chosen after some testing but can probably still be improved.

Further, integrating the developed deep learning model in a separate application to be used in the real world could possibly be an interesting road to follow.

References

[1] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] N. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] C. Barata, M. Celebi, and J. Marques. Melanoma detection algorithm based on feature fusion. In *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, volume 2015, 08 2015.

[4] C. Barata, M. Celebi, and J. Marques. A survey of feature extraction in dermoscopy image analysis of skin cancer. *IEEE Journal of Biomedical and Health Informatics*, PP:1–1, 06 2018.

[5] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[6] T. Brinker, A. Hekler, A. Enk, C. Berking, S. Haferkamp, A. Hauschild, M. Weichenthal, et al. Deep neural networks are superior to dermatologists in melanoma image classification. *European Journal of Cancer*, 119:11–17, 08 2019.

[7] T. Brinker, A. Hekler, A. Enk, J. Klode, A. Hauschild, C. Berking, et al. A convolutional neural network trained with dermoscopic images performed on par with 145 dermatologists in a clinical melanoma image classification task. *European Journal of Cancer*, 111:148–154, 03 2019.

[8] T. Brinker, A. Hekler, A. Hauschild, C. Berking, B. Schilling, A. Enk, S. Haferkamp, et al. Comparing artificial intelligence algorithms to 157 german dermatologists: the melanoma classification benchmark. *European Journal of Cancer*, 111:30–37, 02 2019.

[9] ISIC challenge. Leaderboards - isic 2018. https://challenge2018.isic-archive.com/leaderboards/, 2018.

[10] C. Chatterjee. Basics of the classic cnn. https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add, 2019.

[11] F. Chollet. Building powerful image classification models using very little data. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html, 2016.

[12] F. Chollet et al. Keras. https://keras.io, 2015.

[13] N. Codella, J. Cai, M. Abedini, R. Garnavi, A. Halpern, and J. Smith. Deep learning, sparse coding, and svm for melanoma recognition in dermoscopy images. *Machine Learning in Medical Imaging*, 2015.

[14] N. Codella, V. Rotemberg, P. Tschandl, M. Celebi, S. W. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, H. Kittler, and A. Halpern. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC). *CoRR*, abs/1902.03368, 2019.

[15] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery.

[16] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[17] W. Damsky and M. Bosenberg. Melanocytic nevi and melanoma: unraveling a complex relationship. *Oncogene*, 36(42), 2017.

[18] J. Deng, W. Dong, R. Socher, et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[19] DermnetNZ. Benign melanocytic lesions. https://dermnetnz.org/cme/lesions/benign-melanocytic-lesions/, 2008.

[20] W. A. Dorland. *Dorland's illustrated medical dictionary*. Saunders/Elsevier, 32 edition, 2012.

[21] A. Esteva, B. Kuprel, R. Novoa, J. Ko, S. Swetter, H. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639), 2017.

[22] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[23] D. Frossard. Vgg in tensorflow, 2016.

[24] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

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

[26] J. Hagerty, R. Stanley, H. Almubarak, N. Lama, et al. Deep learning and handcrafted method fusion: Higher diagnostic accuracy for melanoma dermoscopy images. *IEEE Journal of Biomedical and Health Informatics*, PP:1–1, 01 2019.

[27] E. Hale and C. Hanke. Squamous cell carcinoma - the skin cancer foundation. https://www.skincancer.org/skin-cancer-information/squamous-cell-carcinoma/, 2019.

[28] M. Hassan. Vgg16 – convolutional network for classification and detection. https://neurohive.io/en/popular-networks/vgg16/, 2018.

[29] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[31] T. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[32] M. Hoffman. The skin (human anatomy): Picture, definition, function, and skin conditions. https://www.webmd.com/skin-problems-and-treatments/picture-of-the-skin#1, 2014.

[33] K. Holland and J. Marcin. Skin lesions: Pictures, causes, types, risks, diagnosis, and treatments. https://www.healthline.com/health/skin-lesions, 2016.

[34] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[35] National Cancer Institute. Pdq melanoma treatment. https://www.cancer.gov/types/skin/hp/melanoma-treatment-pdq. Accessed: 2020/03/05.

[36] National Cancer Institute. Pdq skin cancer treatment. https://www.cancer.gov/types/skin/hp/skin-treatment-pdq. Accessed: 2020/03/05.

[37] National Cancer Institute. What does melanoma look like? https://www.cancer.gov/types/skin/melanoma-photos, 2011.

[38] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[39] P. Ippolito. Support vector machines. https://pierpaolo28.github.io/blog/blog6/, 2020.

[40] A. Ivakhnenko and V. Lapa. *Cybernetic Predicting Device.* CCM Information Corp, 1965.

[41] J. Karen and R. Moi. Basal cell carcinoma - the skin cancer foundation. https://www.skincancer.org/skin-cancer-information/basal-cell-carcinoma/, 2019.

[42] G. King and L. Zeng. Logistic regression in rare events data. *Political analysis*, 9(2):137–163, 2001.

[43] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[44] J. Koza, F. Bennett, D. Andre, and M. Keane. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*, pages 151–170. Springer Netherlands, Dordrecht, 1996.

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

[46] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[47] L. Maia, A. Lima, R. Pereira, G. Junior, J. Almeida, and A. Paiva. Evaluation of melanoma diagnosis using deep features. In *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–4, 2018.

[48] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[49] T. Mendonça, P. Ferreira, J. Marques, A. Marçal, and J. Rozeira. Ph2 - a dermoscopic image database for research and benchmarking. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, 2013:5437–5440, 07 2013.

[50] P.F. Millington and R. Wilkinson. *Skin*. Biological Structure and Function Books. Cambridge University Press, 2009.

[51] M. Minsky and S. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, Cambridge, MA, USA, 1969.

[52] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[53] T. Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006.

[54] A. Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *NIPS*, 2013.

[55] S. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[58] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[59] S. Ray. Disease classification within dermascopic images using features extracted by resnet50 and classification through deep forest, 07 2018.

[60] A. Romero, C. Gatta, and G. Camps-Valls. Unsupervised deep feature extraction for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(3):1349–1362, 2015.

[61] A. Rosebrock. Imagenet: Vggnet, resnet, inception, and xception with keras - pyimagesearch. https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/, 2017.

[62] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[63] M. Ruela, C. Barata, J. Marques, and J. Rozeira. A system for the detection of melanomas in dermoscopy images using shape and symmetry features. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 5:1–11, 05 2015.

[64] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[65] O. Russakovsky, J Deng, H. Su, J. Krause, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[66] T. Satheesha, D. Satyanarayana, M. Prasad, and K. Dhruve. Melanoma is skin deep: A 3d reconstruction technique for computerized dermoscopic skin lesion classification. *IEEE Journal of Translational Engineering in Health and Medicine*, PP:1–1, 01 2017.

[67] S. Sayad. Decision tree. http://www.saedsayad.com/decision_tree.htm, 2010. Accessed: 2020-02-09.

[68] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[69] W. et al. Sondermann. Prediction of melanoma evolution in melanocytic nevi via artificial intelligence: A call for prospective data. *European Journal of Cancer*, 119:30–34, 2019.

[70] 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*, 15(1):1929–1958, 2014.

[71] T. Srivastava. Introduction to k-nearest neighbors. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/, 2018.

[72] Mayo Clinic Staff. Moles. https://www.mayoclinic.org/diseases-conditions/moles/symptoms-causes/syc-20375200, 2019.

[73] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[74] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[75] P. Tschandl, C. Rosendahl, and H. Kittler. The ham10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5, 03 2018.

[76] F. Xie, H. Fan, L. Yang, Z. Jiang, R. Meng, and A. Bovik. Melanoma classification on dermoscopy images using a neural network ensemble model. *IEEE Transactions on Medical Imaging*, PP:1–1, 12 2016.

[77] F. Yu. A comprehensive guide to fine-tuning deep learning models in keras (part i). https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html, 2016.

[78] Z. Zhou and J. Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.