The People's Democratic Republic of Algeria
Abbès Laghrour University
Faculty of Sciences and Technology
Mathematics and Computer Science Department

(2022-2023)

Master's thesis

Melanoma classification using deep learning

Written by:
HITTA Aymen

Supervised by:
Dr. BARDOU Dalal

Abstract

Melanoma, the deadliest form of skin cancer, has been a growing concern worldwide. Early and accurate detection of melanoma is crucial for improving patient outcomes. In recent years, deep learning techniques have shown promising results in various medical applications, including skin cancer classification. This master's thesis presents a deep learning model that utilizes Convolutional Neural Networks (CNN) and transfer learning to classify melanoma lesions into melanoma and non-melanoma categories. The proposed model builds upon the powerful features learned from pre-trained CNN model, by fine-tuning them using a large dataset of dermoscopy images. The results demonstrate the effectiveness of the model in accurately distinguishing between melanoma and non-melanoma where an accuracy of 89.19 % is achieved.

Keywords: Artificial intelligence, deep learning, transfer learning, convolutional neural networks, skin cancer, melanoma.

Contents

List of Figures

List of Tables

General introduction

Melanoma, a type of skin cancer that originates from melanocytes, has emerged as a significant global public health concern due to its high mortality rate and increasing incidence. The timely and accurate diagnosis of melanoma plays a crucial role in improving patient outcomes and reducing mortality rates. Dermoscopy, a non-invasive imaging technique, has been widely adopted as a valuable tool to assist dermatologists in melanoma diagnosis. However, accurately and consistently interpreting dermoscopic images remains challenging, necessitating the development of automated systems to aid in melanoma classification. In recent years, deep learning algorithms have exhibited remarkable success in various medical applications. Convolutional Neural Networks (CNNs), a prominent deep learning architecture, have demonstrated exceptional performance in extracting meaningful features from images and achieving state-of-the-art results in various image classification tasks. Transfer learning, a technique that utilizes pre-trained models on large-scale datasets, has emerged as a powerful approach to overcome limitations associated with limited training data. This master's thesis aims to address the challenges of melanoma classification by developing a deep learning-based model that combines the strengths of CNNs and transfer learning. The proposed model will be trained on a diverse dataset of dermoscopic images, encompassing a wide range of melanoma and non-melanoma lesions with varying clinical scenarios and skin types. By fine-tuning pre-trained CNN models, the proposed model aims to learn discriminative features relevant to melanoma classification, thereby improving the accuracy and robustness of the classification task. The developed model can serve as a valuable tool to assist dermatologists in the early detection and diagnosis of melanoma, ultimately leading to improved patient outcomes. By automating the melanoma classification process, the model can help reduce subjectivity and variability among clinicians, potentially contributing to more efficient and accurate diagnoses. The subsequent sections of this thesis will provide an introduction to the diagnosis of melanoma, a detailed description of deep learning and CNNs, as well as the methodology employed, including the dataset used, the architecture of the deep learning model, and the training and evaluation procedures.

Chapter 1

Melanoma of the skin

Introduction

Before discussing malignant melanoma, this chapter will provide a concise overview of the human skin and its different layers. As well as some informations and statistics related to skin cancer in general both locally and globally.

1.1 Anatomy of the skin

The human skin is the body's biggest organ, it encircles the entire body and acts as a protective shield against heat, damage, and infection [41]. It is composed of three layers: the epidermis, dermis and Subcutaneous fat layer (hypodermis) [36].

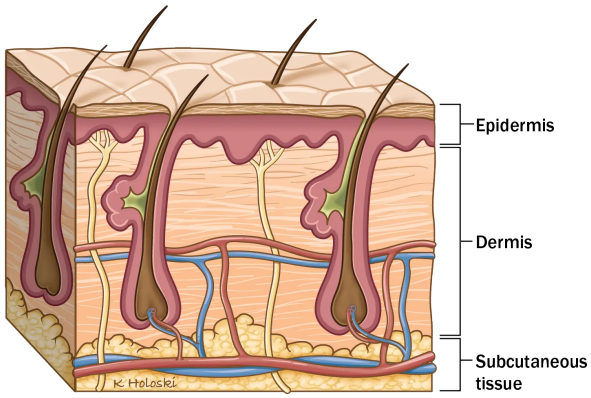


Figure 1.1: Layers of the skin [16]

The lowest layer of skin is the subcutaneous fat layer. It is made up of a web of collagen and fat cells. By serving as a shock absorber, it helps the body retain heat and guards against harm [3].
For the epidermis layer it mainly consist of keratinocytes with the existence of other cells such as melanocytes, Langerhans cells, and Merkel cells. It is organized into four strata, or layers, which are crossed by sweat glands and pilosebaceous units, among other skin appendages [36].
In the other hand, A reticular layer and a papillary layer make up the dermis. It contains the neurovascular supply for the skin. The superficial fascia and subcutaneous fat are located in the subcutaneous tissue beneath the skin. [36].

1.1.1 The epidermis layer

The epidermis is the skin's topmost layer (figure 1.2), it controls the hydration, texture, and color of the skin [36]. The keratinocyte is the main type of cell in the epidermis, and the four epidermal layers show how keratinocytes develop from the deep to superficial layer [36].

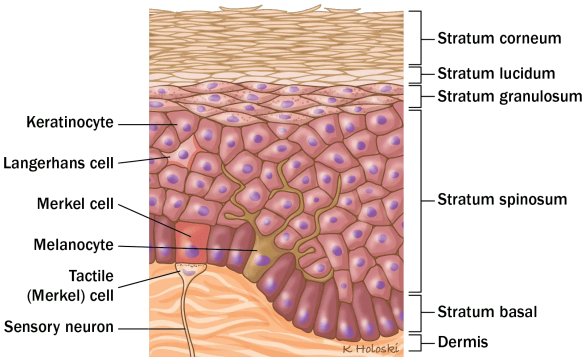


Figure 1.2: The epidermis layer of the skin [16]

1.1.2 The dermis layer

The dermis, which is located beneath the epidermis, is what determines how thick the skin is in different regions [36].
The dermis is further classified as papillary and reticular. The papillary dermis is a loose mixture of fibrocytes, collagen, and blood vessels that lies under the dermal-epidermal junction. Below it is the considerably thicker reticular dermis, which has fewer fibrocytes but a denser collagen accumulation [36].

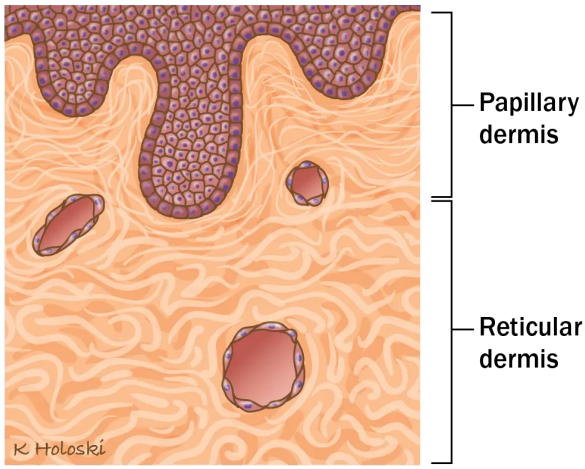


Figure 1.3: The dermis layer of the skin [16]

1.2 Skin cancer

Skin cancer is now the most frequent type of cancer in light skin populations, and its prevalence has reached epidemic proportions [44]. According to recent population-based research from Australia, the incidence rate for basal cell carcinoma in males

is over 2% and 1% for squamous cell carcinoma, with over 50 new occurrences of melanoma per 100,000 people [29].
Melanoma and nonmelanoma skin cancers (NMSC) are the two types of skin cancer. Because of the low probability of metastasis from squamous cell carcinoma (SCC) (Figure 1.4) and the extremely low risk of metastasis from basal cell carcinoma (BCC) (Figure 1.5), these two most frequent cutaneous malignancies are commonly referred to as NMSC [34].

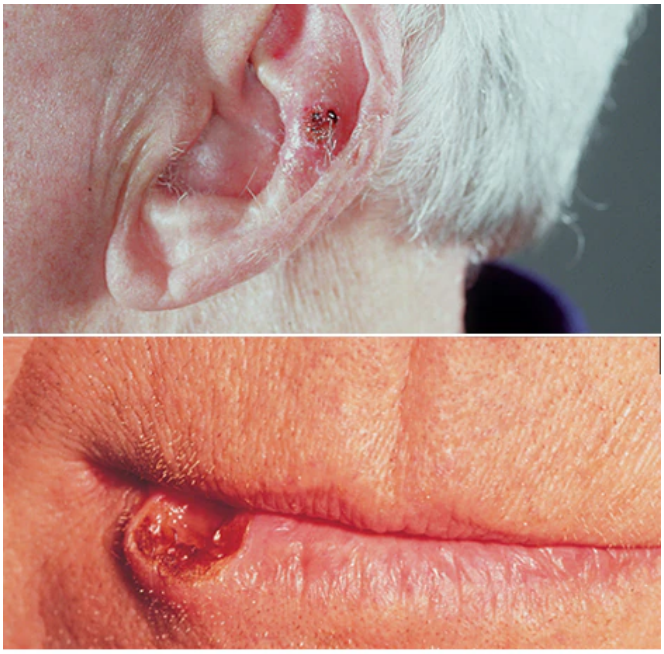


Figure 1.4: Squamous cell carcinoma of the skin [17]

(a) Basal cell carcinoma on light skin

(b) Basal cell carcinoma on dark skin

Figure 1.5: Basal cell carcinoma [4]

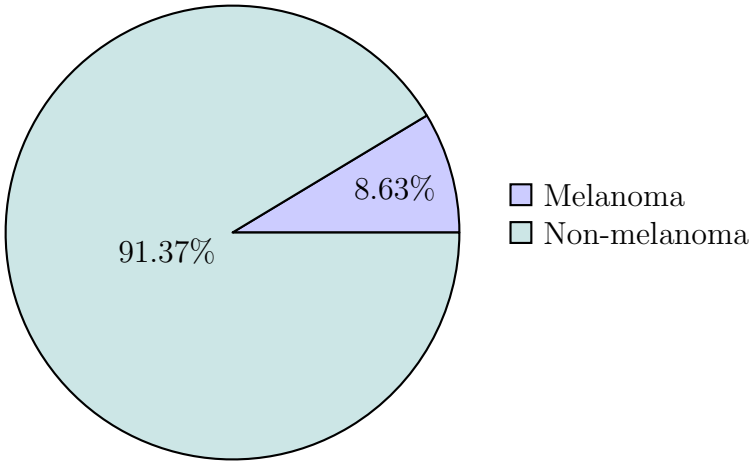
1.3 The state of skin cancer in Algeria

According to the African archive of cancer, the number of cancer incidents recorded at Algiers starting from the year 1993 up to 1997, is as follow:

Cancer	Males	Females	Total number of cases
Breast	26	906	932
Lung	635	73	708
Cervix uteri	/	506	506
Bladder	402	86	488
Skin cancer	314	161	475
Stomach	211	155	366
Rectum and anus	154	133	287
Non-Hodgkin lymphoma	161	109	270

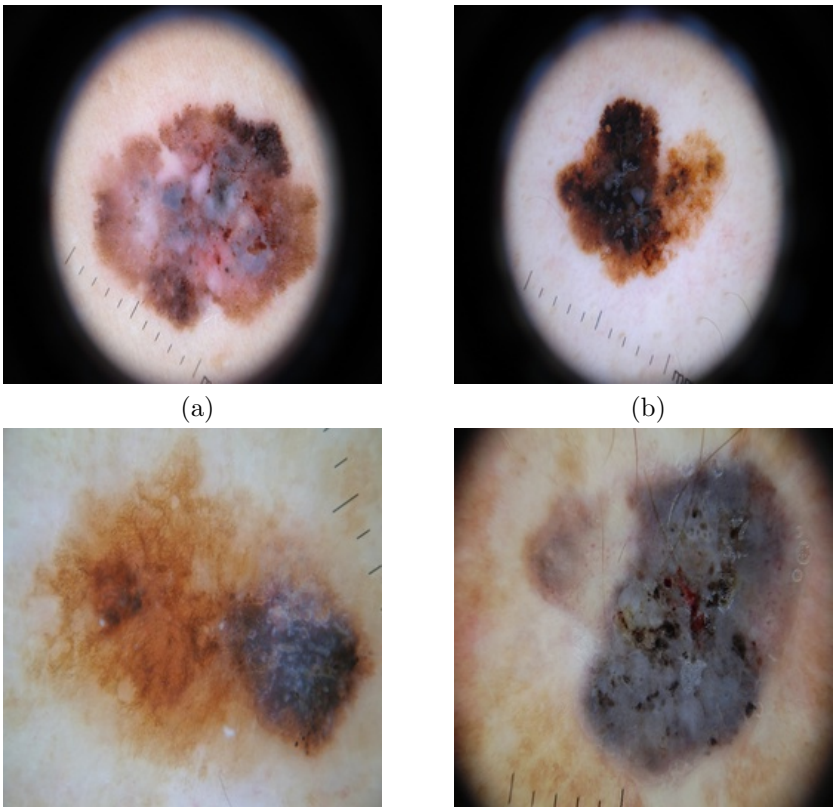
Table 1.1: Cancer incidence at Algiers (1993-1997, age [0-85+]) [9]

With 475 recorded cases, skin cancer is the fifth most common form of cancer in Algeria. Where males makes up to 66.11% of the affected population [9]. Non-melanoma skin cancer is the most frequent form of skin cancer with 91.37% cases, while melanoma skin cancer make roughly 8.63% from the total number of cases [9].



1.4 Melanoma of the skin

Melanoma of the skin (Figure 1.6) is known to be the lesser common yet most fatal form of dermatological cancer mainly due to its ability to spread rapidly to and infect other parts of the human body [34]. In 1991, melanoma of the skin was ranked as the eighth most common type of cancer in the world [31].



(a) (b)

(c) (d)

Figure 1.6: Melanoma of the skin [46, 26, 28]

Like any other type of cancer, early detection is a crucial step in increasing the survival chance for patients with melanoma of the skin. That is, the deeper the tumor penetrates the skin, the lower the chance of survival, and vice versa [31].

1.4.1 Risk Factors

When compared to the general population, people who have one or two of following risk factors have a higher probability of getting melanoma of the skin:

- There is a family history of melanoma [40].
- Having blond or red hair [40].
- The presence of prominent freckling on the upper back [40].
- Prior to the age of 20, the person in question had had three or more extremes sunburns [40].
- Three or more years of outdoor summer work experience as a teenager [40].

1.4.2 Symptoms

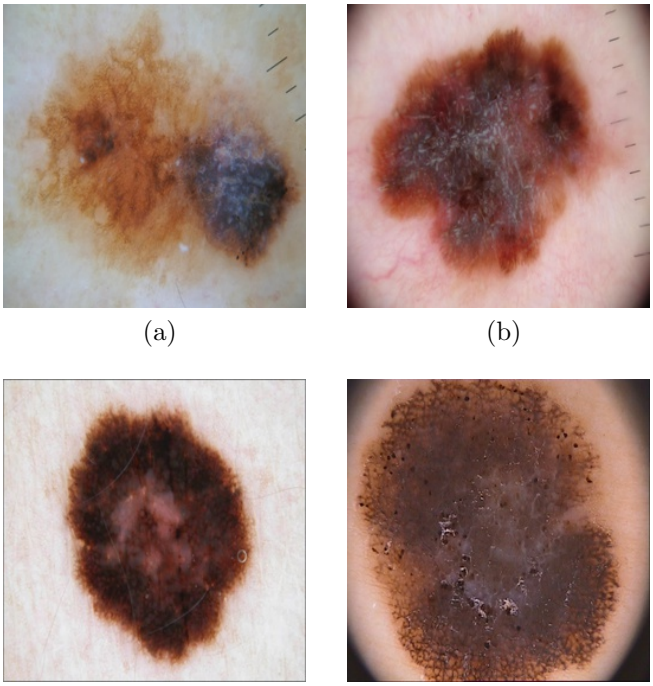
Melanoma frequently appears as an irregularly bounded, pigmented macule with a variety of colors ranging from tan to brown to jet-black, although it can also be equally colored [34]. Scientists have came-up with a guideline that can help with identifying early melanoma, the guideline was named The ABCDs of melanoma [31]:

A for asymmetry: The majority of early lesions develop at an uneven rate, creating an asymmetric disjointed pattern [40].

B for border inconsistency: An inconsistent border is also a result of the unequal growth rate [40].

C for color variegation: New tones of black, as well as pale and dark brown, are also brought on by irregular growth [40].

D for diameter: Lesions having diameters higher than 6 mm ought to be suspected of being melanoma [40].



(a) (b)

(c) (d)

Figure 1.7: The ABCDs of melanoma [46, 26, 28]

1.4.3 Types of Melanoma

Their exist four kinds of melanoma skin cancer: acral lentiginous, nodular, lentigo maligna, and superficial spreading [27].

- **Acral lentiginous melanomas** are distinguished by a lentiginous (radial) development phase that progresses over several months or years to a dermal (vertical) invasive stage. This melanoma variant is most likely the most prevalent form of the disease among people of dark skin color [27].
- **Nodular melanomas** are distinguished by a lack of radial growth, vertical growth from tumor origin, epithelioid or spindle tumor cells, epidermal thinning over the developing tumor nodule, and cellular host immune response with specific areas of regression [27].

- **Lentigo maligna melanomas** progress slowly from a radial to a vertical development phase. It is an actinically generated lesion that affects the sun-damaged skin of elderly people with fair complexions. It has a lentiginous radial growth component with variable melanocytic dysplasia, a vertical growth component that appears late in the evolution of the radial component, spindle cells that are diffusely arranged in the basal layer of the epidermis in the radial component and compactly aggregated to form a nodule in the vertical component, and an atrophie epidermis with effaced rete ridges [27].
- **Superficial spreading melanomas** are characterized by the following: Pagetoid or nevocytic radial growth pattern with uniform melanocytic dysplasia, a vertical growth component in which the tumor evolves as an expanding nodule in the papillary dermis for a period of time, epithelioid tumor cells, stratum malpighii hyperplasia in the radial component, and a marked host immune response with focal areas of regression [27].

1.5 Melanoma statistics

Since 1960, the annual mortality rate from melanoma of the skin has increased by roughly 2% [40]. Additionally, a study that was conducted in 2008 has estimated that there were nearly 200,000 new cases of melanoma worldwide and around 46,000 deaths caused by it, making it the leading cause of skin cancer related deaths [30]. It is estimated that more than 7,990 deaths caused by melanoma will be recorded in 2023 [11].

1.5.1 By age

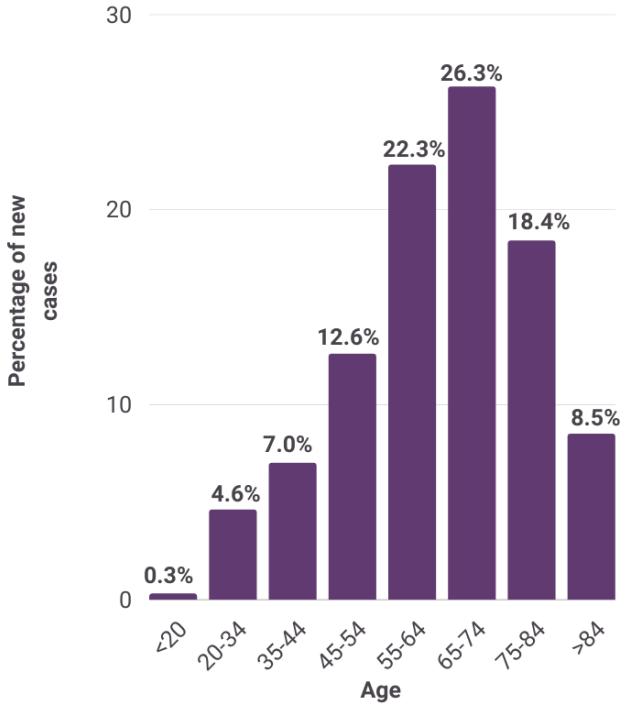


Figure 1.8: Percentage of new cases by age [11]

While it is less common in teenagers and children (only 0.3 percent of confirmed cases), melanoma of the skin is most typically found in adults aged 65 to 74 (up-to 26.3 percent of total diagnosed cases), with a median age of diagnosis of 66 years old.[11].

1.5.2 By sex/race

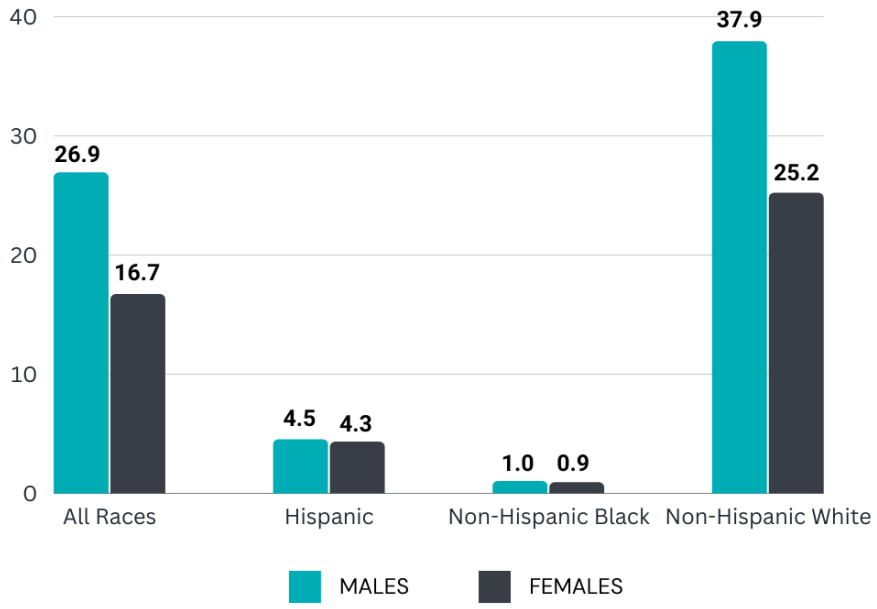


Figure 1.9: Rate of New Cases per 100,000 Persons by Race/Ethnicity and Sex: Melanoma of the Skin [11]

Sex

Both in Algeria and worldwide, men are more likely than women to develop melanoma [9] [11]. In general, out of every 100,000 people, about 26.9 men had melanoma, whereas only 16.9 of the new cases were women [11].

Race

Melanoma affects the non-Hispanic White population the most. Following that is the Hispanic population. Non-Hispanic blacks, on the other hand, are the least affected by melanoma [11].

1.5.3 The 5 years survival rate

Almost 93.5% of patients survive melanoma within 5 years or less of being diagnosed with it [11].
Malignancy stage at diagnosis, which refers to the size of a malignancy in the body, dictates treatment options and has a significant impact on survival time [11]. In general, cancer is considered localized if it is discovered solely in the area of the body where it began (also known as stage 1) [11]. The stage is regional or distant if it has spread

to another portion of the body [11]. The earlier melanoma of the skin is detected, the greater the chances of survival five years later [11]. Skin melanoma is identified at the local stage in 77.6% of cases [11] and 35.1% chance when being distant. The 5-year relative survival rate for localized melanoma is 99.6% [11].

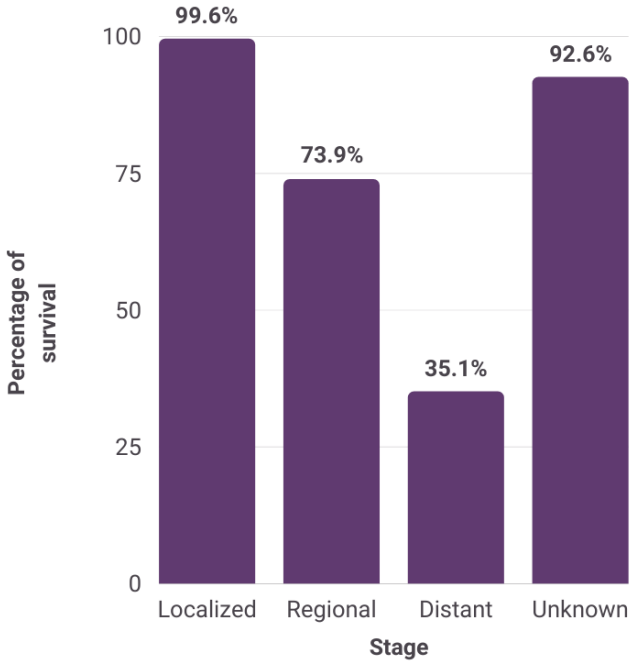


Figure 1.10: Skin Melanoma 5-Year Relative Survival Rates by Stage [11]

1.5.4 Conclusion

In recent decades, the world has witnessed an exponential increase in new melanoma of the skin cases that has reached an epidemic levels. Fortunately, thanks to early detection, that increase in melanoma cases comes with an increase in survival rate (5 years survival rate) making it less fatal then before.

Chapter 2

Artificial intelligence

2.1 Introduction

When asked about artificial intelligence most people will picture a human looking-like robot, that can understand and communicate with us clearly in human voice, that can perform the basic tasks that regular humans can do such as vehicles driving, house cleaning, ...etc. This chapter will attempt to demystify the magic behind AI and how it works.

2.2 Artificial intelligence (AI)

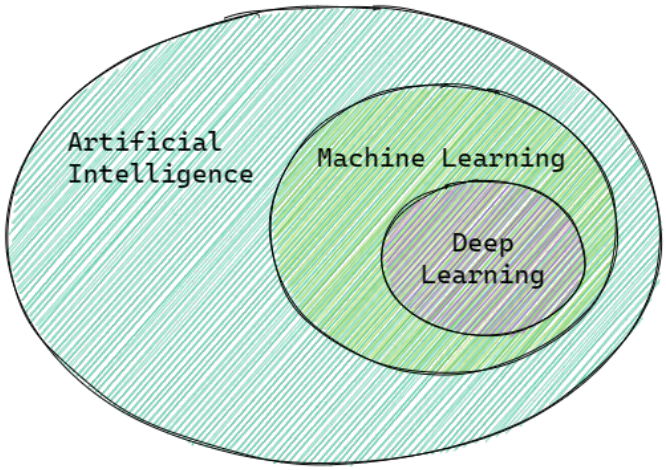


Figure 2.1: AI and some of it sub-fields

The term "artificial intelligence" (AI) today refers to a vast array of subfields, from the general (learning and perception) to the specialized (playing chess, proving mathematical theorems, writing poetry, driving a car through a congested street, and detecting diseases).. [42].
Scientists still cannot agree on a single definition of artificial intelligence, but it is reasonable to group the various concepts put forth during the course of the years into four categories: thinking humanly, thinking rationally, acting humanly, and acting rationally. When these components come together, they can create a condition that is as

like to human intelligence as is conceivable.[42] Listed below are some of the previously mentioned definitions:

Thinking humanly

> "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning ..." (Bellman, 1978) [42]

Thinking rationally

> "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992) [42]

Acting humanly

> "The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) [42]

Acting rationally

> "AI ... is concerned with intelligent behavior in artifacts." (Nilsson, 1998) [42]

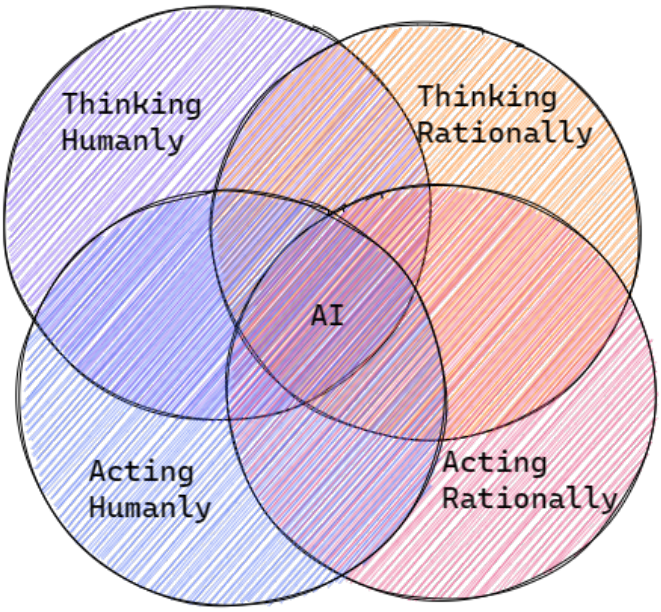


Figure 2.2: Components of AI

2.2.1 Can machines learn?

To answer this question, in 1950, Alan Turing (fig 2.3) who is known as the founding father of artificial intelligence and of modern cognitive science [2] proposed the **Turing Test** to establish a sufficient practical definition of intelligence [47]. Basically, the

test consists of a computer passes the test if a human interrogator cannot distinguish whether the written responses are from a person or from a computer after posing some written questions [42]. Turing's test intentionally excluded a physical encounter between the interrogator and the computer, owing to the fact that physical emulation of a person is not required for intelligence [42].
To pass such a test, the computer would need to have the following abilities:

- Conversing effectively in human language [42].
- Storing what it has learned or heard [42].
- Answering inquiries and making new deductions based on the data that has been saved [42].
- Adjusting to new situations while spotting patterns and drawing conclusions from them [42].

Each and every one of the previously stated abilities are possible to attain respectively through Natural Language Processing, Knowledge Representation, Automated Reasoning and Machine Learning [42].

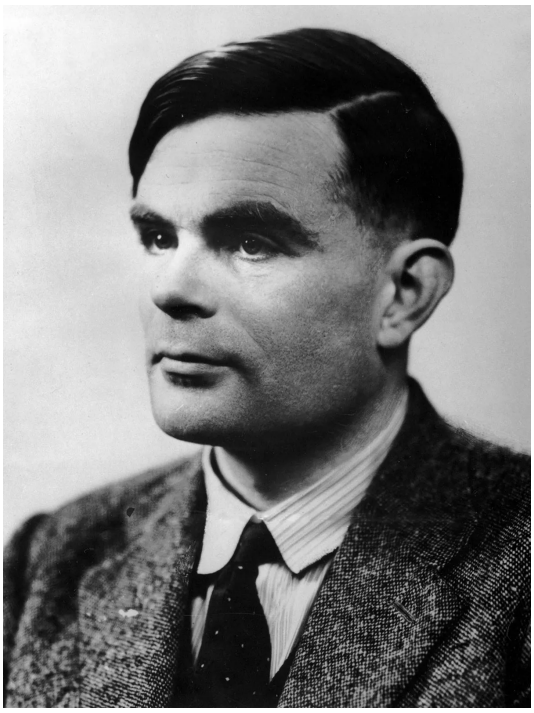


Figure 2.3: Alan Turing [2]

2.3 Machine learning (ML)

The science (and art) of programming computers to learn from data is known as machine learning [32].

> "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." (Tom Mitchell, 1997) [32]

Example :

Given samples of spam emails (marked by users) and examples of ordinary emails (often known as "ham"), the program could gain knowledge to mark them as spam. [32]
In this scenario, the task T is to identify spam in new emails, the experience E is the training data, and the performance measure P needs to be established; one option is to utilize the percentage of emails that are appropriately classified. Accuracy is the name of this specific performance metric, which is frequently applied to classification tasks. [32]

2.3.1 When to apply machine learning

Machine learning is excellent at:

- Issues where current solutions need for extensive fine-tuning or lengthy lists of rules (a ML model can frequently cut down on code and outperform the conventional approach) [32].
- Highly complicated problems for which traditional methods fail to produce good results (but for which the finest ML approaches may be able to discover a solution) [32].
- Unpredictable and changing environments (a ML system can be quickly updated by retraining on new data) [32].
- Gaining knowledge regarding complicated issues and an immense amount of data [32].

2.4 Deep learning

Because deep learning has evolved gradually over the past ten years, it has been difficult to define for many people. According to one helpful definition, deep learning involves **greater than two-layer neural networks**. [39]

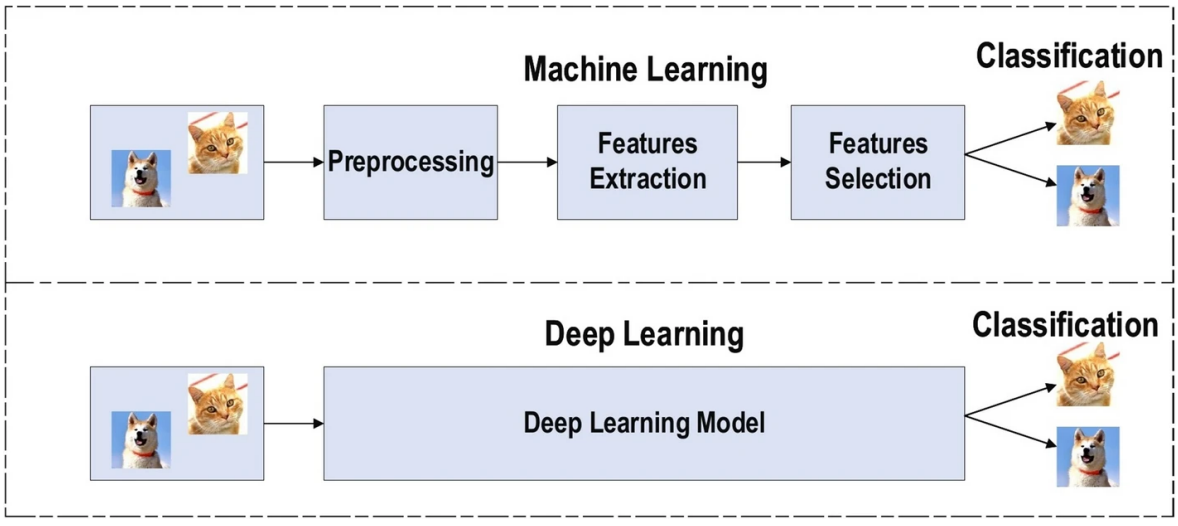


Figure 2.4: The difference between ML and DL [24]

More specifically, it is a subset of machine learning, which is itself a sub-field of AI. It is defined as neural networks with a large number of parameters and layers that operate in one of four basic network architectures: (Pretrained unsupervised networks, Recurrent neural networks, Convolutional neural networks, Recursive neural networks). [39]

2.4.1 When to apply deep learning

DL can help with the following issues:

- Cases in which there are no human specialists accessible [24].
- Cases in which individuals are incapable of clarifying assessments made utilizing their knowledge (such as in speech recognition, language understanding, and medical decisions) [24].
- Situations (price prediction, stock preference, weather prediction, and monitoring) where the answer to the problem changes over time [24].
- Situations when answers need to be modified based on particular circumstances (personalization, bio-metrics) [24].
- Situations in which the complexity of the problem is so great that it defies our little capacity for thinking (sentiment analysis, ranking web pages) [24].

2.4.2 Neural networks

Neural networks are a type of computing model that exhibits several characteristics of the human brain, including the parallel operation of numerous small units without a single central control. The main mechanism for long-term storage of knowledge in neural networks is the weights between the units. The neural network mostly acquires new knowledge by changing the weights. [39]
The feed-forward multi-layer neural network is the most popular and easiest to comprehend neural network. It consists of a single output layer, one or more hidden layers, and an input layer. The number of neurons in each layer can vary, and each layer is completely coupled to its neighboring layer. [39]

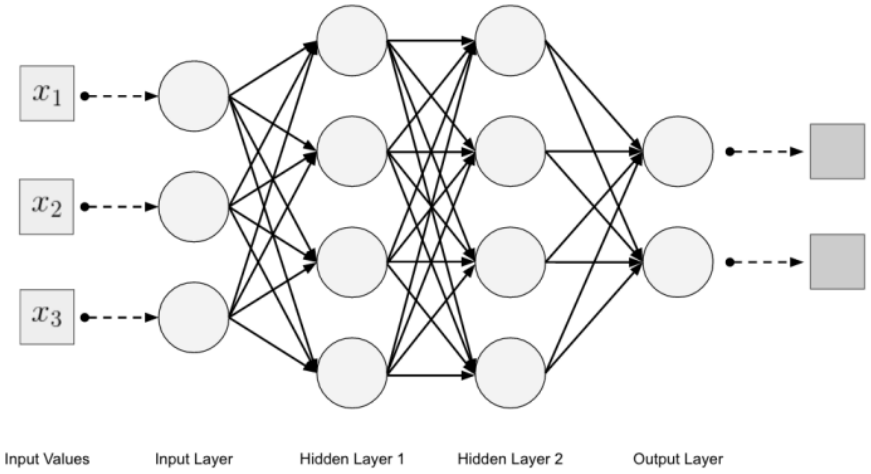


Figure 2.5: Multi-layer Feed-Forward Neural Network [39]

2.5 Different approaches of learning

Using a set of assessments to identify an underlying process is the fundamental tenet of learning from data. It is challenging to confine such a vast premise to a single framework. As a consequence, various learning paradigms have developed to address various circumstances and presumptions. [21]

2.5.1 Supervised learning

We are in the supervised learning scenario when the training data incorporates explicit examples of what the right output should look like for specific inputs [21].
Classification is a classic supervised learning task. A great illustration of this is the spam filter, which requires understanding how to categorize new emails after being taught with several examples of emails along with their correct class (spam or ham). [32]

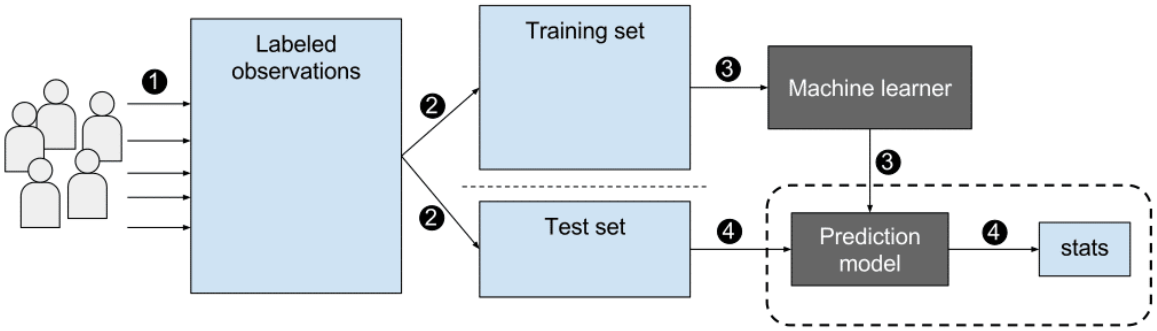


Figure 2.6: Supervised Learning [43]

2.5.2 Unsupervised learning

Unsupervised learning could be thought of as the process of discovering patterns and structure on one's own in input data [21]. In other words, unsupervised learning is when a system tries to learn on its own without a tutor using training data that has not been labeled [32].

For instance, the system can recognize books that have similar features and group them together in one category without naming that category if given the task of categorizing a set of books into subjects using only generic properties of the individual books [21].

Anomaly detection (fig 2.7) is yet another major unsupervised task; examples include spotting fraudulent credit card transactions, finding production flaws, or automatically deleting outliers from a dataset before passing it to another learning model. During training, the system is primarily exposed to regular cases so that it may learn to recognize them. Once it has done so, it is able to determine whether a new instance resembles a typical one or is more likely to be an anomaly. [32]

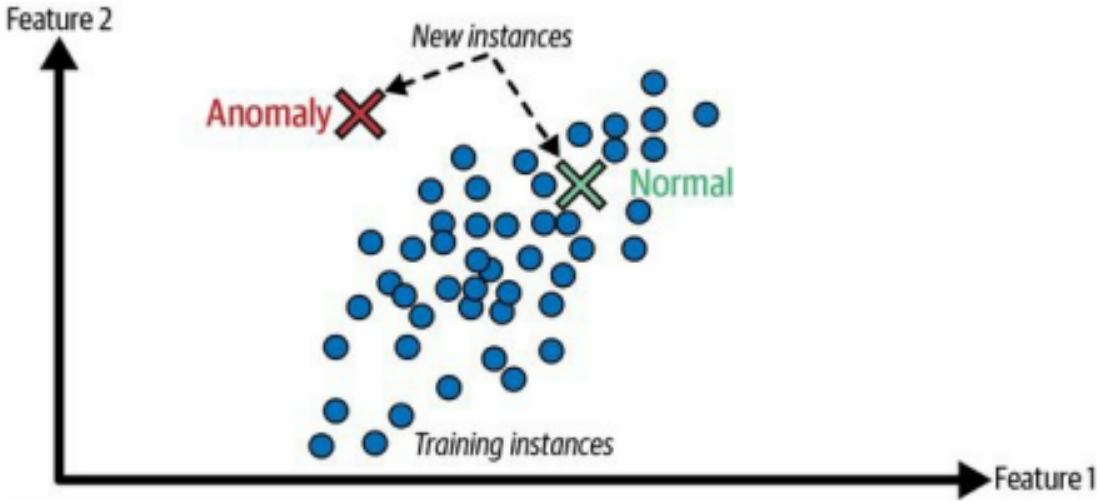


Figure 2.7: Anomaly detection using unsupervised learning [32]

2.5.3 Semi-supervised learning

In semi-supervised learning, The procedure for learning is based on semi-labeled data. This method has the benefit of requiring the least amount of labeled data possible. However, this approach has some drawbacks, including the potential for inaccurate decision-making due to irrelevant input features included in the data used for training. Semi-supervised learning is known to be the best option for text document classification tasks because it is challenging to acquire a large number of labeled text documents. [24]

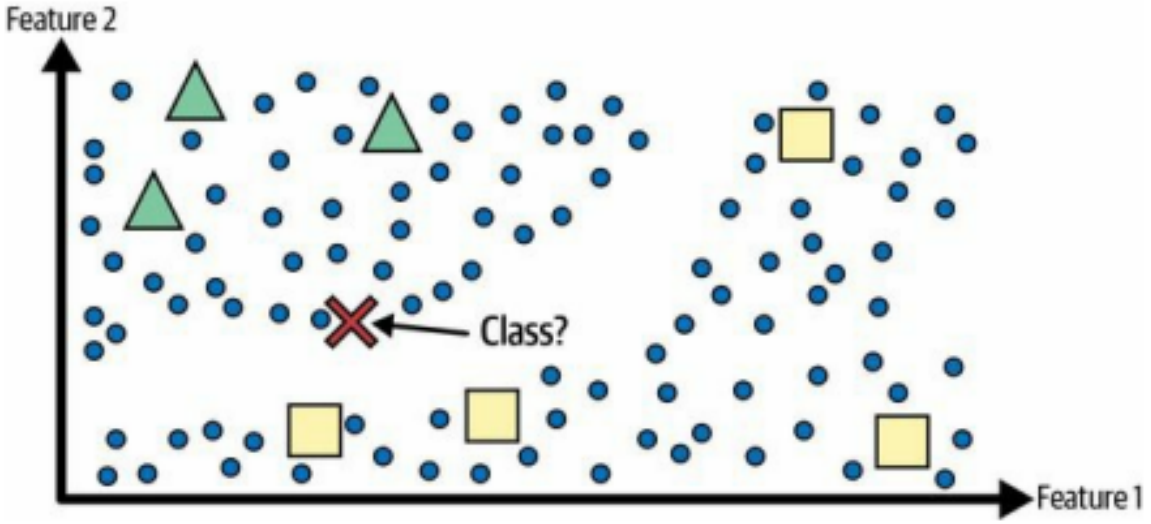


Figure 2.8: Semi-Supervised Learning [32]

The majority of semi-supervised learning methods are hybrids of unsupervised and supervised techniques. A clustering method, for example, may be utilized to organize similar instances together, subsequently each unlabeled instance could be tagged with whatever is the most prevalent label in that cluster. Once the entire dataset has been labeled, any algorithm that uses supervised learning can be applied. [32]

2.6 How do neural networks learn?

The majority of datasets show a substantial correlation between certain attributes and labels (for example, a relationship between living space and a home's sale price). By generating a rational prediction based on the provided data and weights, neural networks understand these correlations without being aware of them, and then, the accuracy of the outputs is measured. [39]
The network is rewarded for accurate predictions and punished for inaccurate ones by the loss functions used in optimization methods like stochastic gradient descent (SGD). SGD steers the network's parameters to avoid producing poor predictions and toward accurate ones. [39]

2.7 Activation functions

In neural networks, activation functions are used to compute the weighted sum of input and biases, which is then utilized to determine whether or not a neuron can activate. In most cases, gradient descent is used to alter the supplied data before producing an output for the neural network that comprises the parameters from the data. [37]

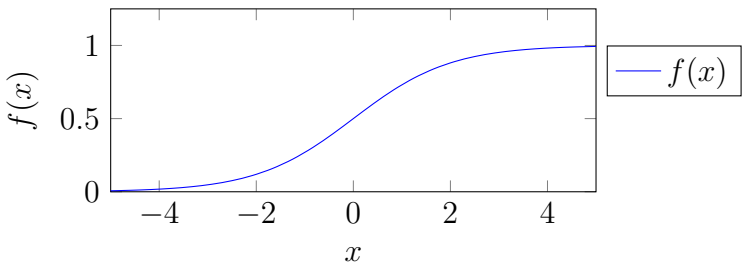
2.7.1 Sigmoid function

$$f(x) = \left(\frac{1}{(1+e^{-x})}\right) \qquad [37]$$

A Sigmoid function is a non-linear activation function mostly used in feed forward neural networks [37]. It is a tool that transforms independent variables with nearly limitless ranges into straightforward probabilities between 0 and 1, with the majority of its output being extremely close to either 0 or 1 [39]. For each class, Sigmoid produces an independent probability [39].

With its primary benefit of being simple for anyone to comprehend, it has been effectively used in binary classification problems, modeling logistic regression tasks, along with other neural network disciplines. [37]

Sharp damp gradients during backpropagation from deeper hidden layers to the input layers, gradient saturation, sluggish convergence, and non-zero centered output are some of the key disadvantages of the Sigmoid AF, which causes the gradient updates to propagate in various directions. In order to address those drawbacks caused by the Sigmoid function, researchers have came up with other activation functions such as hyperbolic tangent (tanh) function. [37]

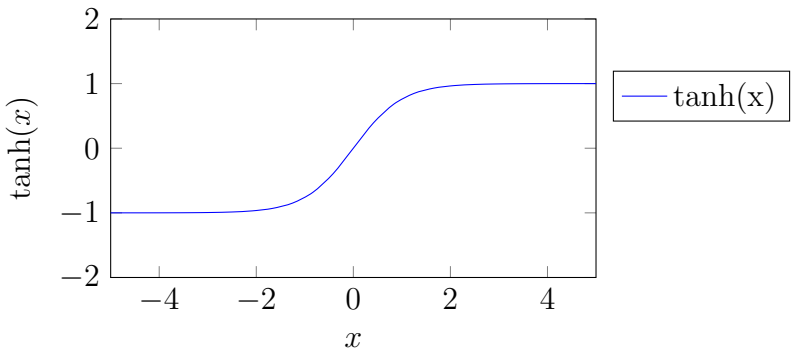


Sigmoid activation function [37]

2.7.2 Tanh function

$$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right) \qquad [37]$$

Another type of AF utilized in deep learning is the hyperbolic tangent function (tanh) which has the benefit of dealing with negative numbers more naturally [37, 39]. Due to its superior efficiency during training, the tanh function has replaced the sigmoid function as the favored activation function for multi-layer neural networks [37]. Tanh's normalized range, in contrast to the Sigmoid function, is from -1 to 1 [37, 39]. It's used primarily in recurrent neural networks for tasks involving speech detection and natural language processing [37].
The tanh function has the peculiarity that it is able to obtain a gradient of 1 when the input value is 0. As a result, the computational process of the tanh function results in some dead neurons. The activation weight is infrequently utilized in the case of dead neurons due to 0 gradient. The rectified linear unit (ReLU) activation function was created as a result of more research into activation functions to address the tanh function's drawback. [37]



Tanh activation function [37]

Hard Tanh

$$f(x) = \begin{cases} 1, & \text{if } x > 1 \\ x, & \text{if } -1 \leq x \leq 1 \\ -1, & \text{if } x < -1 \end{cases} \qquad [37]$$

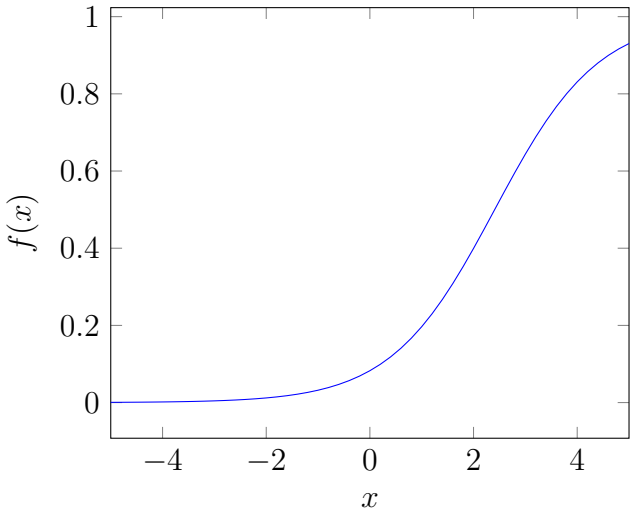
The Hardtanh is a less expensive and more computationally efficient form of tanh [37]. Anything greater than 1 is converted to 1, and anything less than -1 is converted to -1 which result to an even more reliable activation function with a constrained decision boundary [39].

2.7.3 Softmax function

$$f(x) = \left(\frac{e^{x_i}}{\sum_j e^{x_j}} \right) \qquad [37]$$

The Softmax function is yet another kind of activation function utilized in neural computing. It's used to generate a probability distribution from a set of real numbers. The Softmax function returns a range of values between 0 and 1, with the sum of the probability equal to 1. [37]
The Sigmoid is utilized for binary classification whereas the Softmax is mostly applied

in multivariate classification jobs, which is the fundamental distinction between the Sigmoid and Softmax [37].



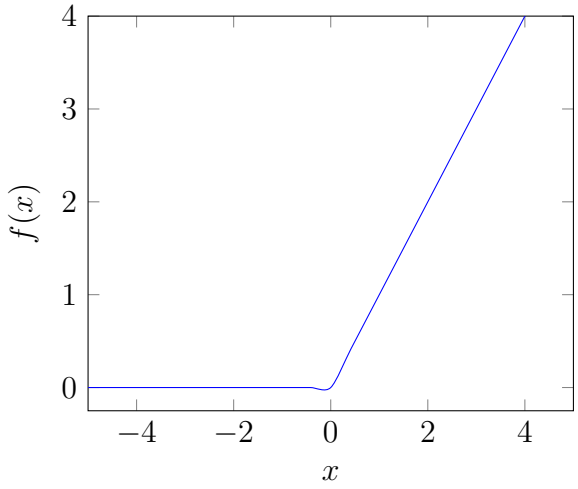
Softmax activation function [37]

2.7.4 Rectified Linear Unit (ReLU) Function

$$f(x) = max(0, x) \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \qquad [37]$$

The ReLU has been proven to be the most successful and extensively utilized activation function [37]. In deep learning, it exceeds the Sigmoid and Tanh activation functions in terms of performance and generalization [37]. And because it represents a nearly linear function, it retains the features of linear models that make them easier to optimize using gradient-descent approaches [37].

The fundamental advantage of utilizing rectified linear units in computation is that they ensure faster calculation because they avoid calculating exponentials or divisions, resulting in increased total computation performance [37].
However, the ReLU has a problem in that it quickly overfits when compared to the sigmoid function, despite the fact that the dropout approach was used to lessen the impact of the overfitting of ReLUs enhanced deep neural network performance [37].



Relu activation function [37]

2.8 Loss functions

Loss functions quantify how near a certain neural network is to the optimum state it is training toward. The concept is straightforward, considering the error in the network's predictions, we compute a metric. We then average these mistakes over the entire dataset to get a single number that represents how close the neural network is to its ideal. [39]
A loss function is denoted as:

$$h_{w,b}(X) = Y \qquad [39]$$

h The loss function used to calculate the error margin.

w The weights vector.

b The bias.

X The input vector.

Y The predicted value.

2.8.1 Cross-Entropy function

Cross-Entropy function is utilized mostly when measuring how well a convolutional neural network perform. When applied it returns the probability $p \in \{0, 1\}$. It makes use of the softmax function in the output layer to generate a prediction within the range of probabilities. [24]

$$h(p, y) = -\sum_{i} y_i log(p_i) \quad where \quad i \in [1, N] \qquad [24]$$

2.8.2 Mean square error function

In problems like regression, this function is often put to use. It is also known as the euclidean loss function. [24]

$$h(p, y) = \frac{1}{2N} \sum_{i=1}^{N} (p_i - y_i)^2 \qquad [24]$$

2.8.3 Hinge function

This function is frequently used in binary classification tasks. It is also used by support vector machines (SVM) [24, 39].

$$h(p, y) = \sum_{i=1}^{N} max(0, m - (2y_i - 1)p_i)^2 \qquad [24]$$

2.9 Issues when training deep neural networks

Even though neural networks have a strong reputation for having the ability to almost perfectly learn everything, there are still many obstacles to overcome before neural networks can be trained to perform at this level. These difficulties are mostly due to a number of practical training issues, the most significant of that being overfitting. [22]

2.9.1 Overfitting

Overfitting signifies the fact that fitting a model to a specific training dataset does not ensure that it will perform well on unknown test data, even if the model precisely predicts the desired results on the training data. In other words, there is always an accuracy disparity between training and test data, which is especially high when the models are complicated and the data set is small in size. [22]
Some of the solutions that could be used to address this problem are:

Regularization: Because an excessive amount of parameters leads to overfitting, it is reasonable to constrain the model to use lesser non-zero parameters [22].

Early Stopping: Another frequent type of regularization is early stopping, which terminates the gradient descent after only a few iterations. Holding out a portion of the training data and subsequently evaluating the model's error on the held-out set is one method for determining the stopping point. When the error on the set begins to climb, the gradient-descent technique is halted. [22]

Ensemble Methods: To improve the model's generalization capacity, a variety of ensemble methods are applied. Dropout and Dropconnect are two of these methods. In many real-world circumstances, these strategies can be used with several neural network topologies to achieve an additional accuracy enhancement of roughly 2%. [22]

Figure 2.9: An example of overfitting

2.9.2 Computational difficulties

Even though Advances in hardware technology, such as Graphics Processor Units (GPUs), have aided significantly in recent years, the amount of time necessary to train the network remains a significant hurdle in building neural networks, it is common for neural networks to take weeks to train [22].
GPUs are highly specialized hardware processors that can dramatically accelerate the kind of operations that are frequently utilized in neural networks [22].

2.10 Convolutional neural network (CNN)

Convolutional Neural Network are designed to handle data that is presented in the form of several arrays, such as a color image made up of three 2D arrays representing the pixel intensities for the three color channels. Numerous data modalities take the shape of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. Local connections, shared weights, pooling, and the usage of several layers are the four main concepts of CNNs. [35]

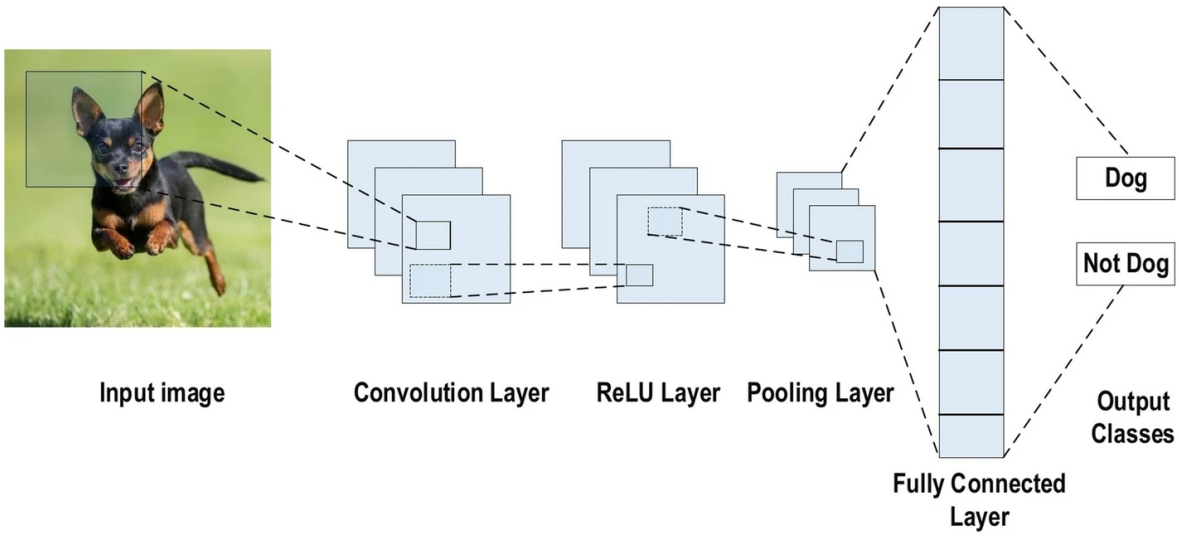


Figure 2.10: An example of a CNN in an image classification task [24]

2.10.1 The architecture of CNNs

The overall architecture of CNN can be broken down into three different components:

1. **Input layer:** The input layer takes three-dimensional data typically in the spatial form of the image's dimensions (width × height) and has a depth that corresponds to the image's color channels, which are typically 3 for RGB (Red, Green, Blue) color channels. [39]

2. **Feature extraction layers:** Mainly consists of the convolution and the pooling layers. These layers identify various features in the photos and build higher-order features one at a time. This strongly relates to the continuing topic of deep learning, where features are acquired automatically rather than manually designed. [39]

3. **Classification layers:** Composed one or more fully connected layers to generate class probabilities or scores using the higher-order features. Each and every neuron in the preceding layer is fully linked to this layer. [39]

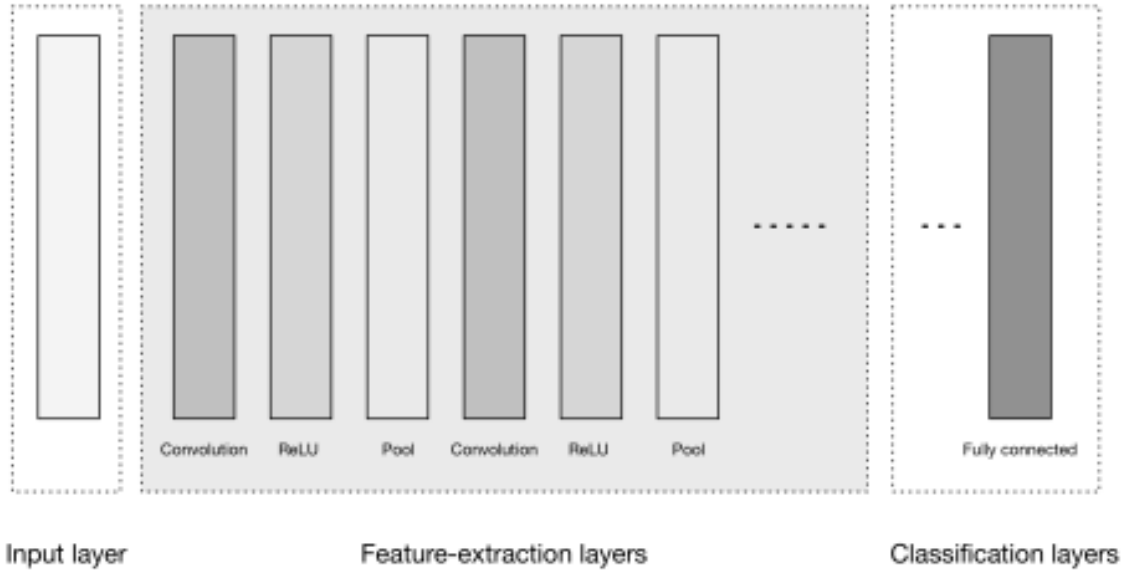


Figure 2.11: A general architecture of CNN [39]

The input layers

The input layers are the component that receive and store the image's input data to be processed in the network. This input data describes the dimensions of the width, height, and total number of channels. Usually, three channels are used to represent the colors in RGB for each pixel.[39]

The feature extraction layers

The most important component of a convolutional neural network is the convolutional layers. They are made up of a series of convolutional filters (known as kernels). The input image is convolved with the aforementioned filters to form the output feature map, which manifests as N-dimensional metrics. [24]

The kernel is described by a matrix of discrete integer values. Each value is referred to as the kernel weight. At the start of the CNN process of training, random numbers get chosen to act as kernel weights. Furthermore, there are numerous approaches for initializing the weights. These weights are then modified at each training epoch, allowing the kernel to learn to extract key features. [24]

Convolutional Operation: A convolution is a mathematical process that describes a rule for combining two distinct sets of information. It is known as a CNN's feature detector. A convolution's input can be plain data or even a feature map produced from another convolution. It is sometimes regarded as a filter that occurs when the kernel filters the data being provided for specific types of information. For instance, a border kernel allows only information near the border of an image to pass through. [39]

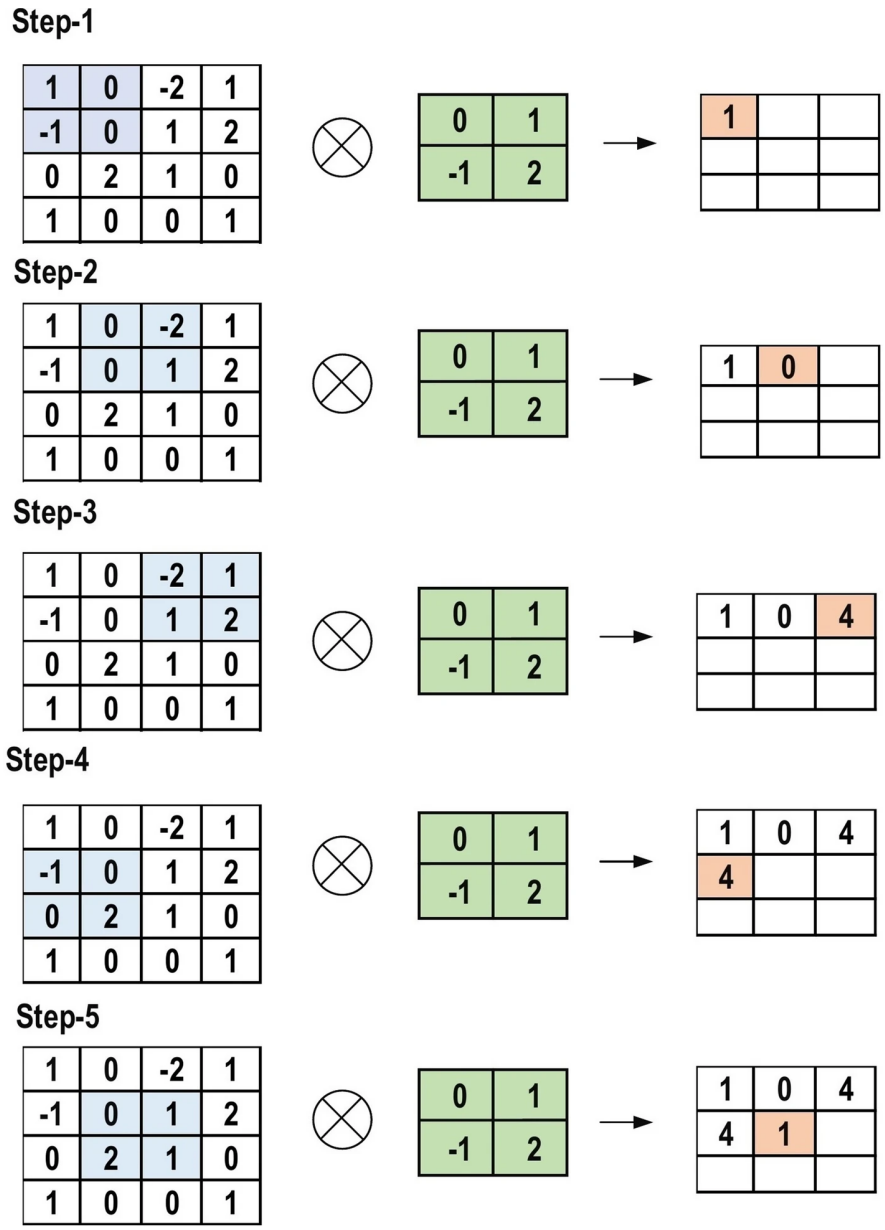


Figure 2.12: An example of convolution [24]

The figure shows how the kernel moves across the input data to generate the convoluted feature data. The kernel gets multiplied by the given input data values within its constraints at each step, resulting in a single item in the resultant feature map. In practice, if the feature we're seeking for is discovered in the input, the output is huge. [39]

The pooling layer's primary objective is to subsample the feature maps. These maps are created by using convolutional operations. In other words, this method condenses large-scale feature maps into smaller feature maps. At the same time, it keeps a majority of the dominating features in every stage of the pooling process. Before the pooling process, both of the stride and the kernel will be size-assigned in the same way as the convolutional operation is. The most common and widely used pooling strategies are max, min, and global average pooling. [24]

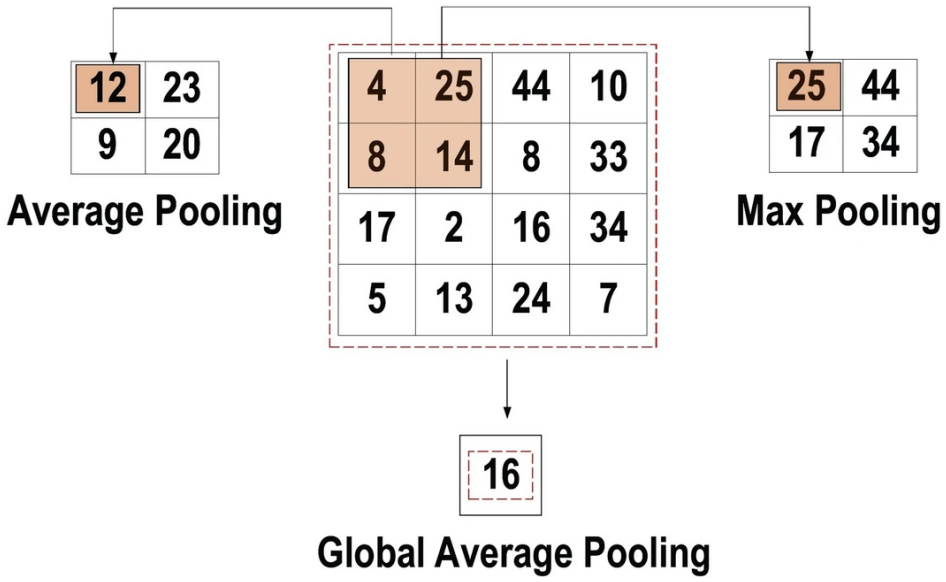


Figure 2.13: The kinds of the pooling operations [24]

The classification layers

This layer is typically found near the last point of each CNN architecture. Each neuron in this layer is connected to every one of the neurons in the previous layer, using the Fully Connected (FC) technique. It serves as the convolutional neural network classifier. As a feed-forward ANN, it follows the basic method of the standard multiple-layer perceptron neural network. [24]

2.10.2 Advantages of CNNs

The following are the advantages of utilizing convolutional neural networks over other regular neural networks in the computer vision field:

- The weight sharing feature, which decreases the amount of trainable network parameters and so allows the network to improve generalization and avoid over-fitting, is one of the major reason for going with CNN [24].
- CNN makes large-scale network construction significantly simpler than other neural networks [24].
- Training the feature extraction layers as well as the classification layer at the same time results in a model output that is both highly ordered and heavily dependent on the extracted features [24].

2.10.3 Application fields of CNNs

The following are some of the fields in which CNNs are mostly used:

Image classification

For a long time, CNNs have been used in image classification. CNNs outperform other approaches in terms of classification accuracy on big scale datasets due to their capacity to learn features and classifications simultaneously. [33]

Object detection

Object detection has long been a significant problem in the field of computer vision. In general, the issues stem primarily from determining how to reliably and quickly pinpoint objects in pictures or video frames. CNNs have been used for detection since the 1990s. However, progress in CNN-based object detection was modest before 2012 due to a shortage of training data and restricted processing capabilities. Since 2012, the overwhelming success of CNNs in the ImageNet challenge has reignited interest in CNN-based object detection. [33]

Natural language processing

While recurrent neural networks are the primary method for learning algorithms using text sequences, convolutional neural networks have grown in popularity in recent years. CNNs do not appear to be a natural fit for text-mining tasks at first glance. First, regardless of where they're situated in the image, image shapes are evaluated in the same way. This is not the situation with text, where the placement of a word in a phrase seems to impact quite a deal. Second, in an image, surrounding pixels are frequently highly similar, yet neighboring words in prose are nearly never the same. Despite these discrepancies, convolutional network-based systems have showed better performance in recent years. [22]

2.10.4 Transfer learning

When trained on images, present-day deep neural networks all have a surprising tendency to capture first-layer features that have similarities with either Gabor filters as well as color blobs. It is referred to these first-layer features as universal since they may be found on the first layer regardless of the precise cost function and raw picture dataset. On the flip side of the hand, it is known that the features obtained by a trained network's final layer must heavily rely on the selected task and dataset, hence the final layer of features is referred to as specific. [49]
By transferring knowledge from a similar domain, transfer learning is utilized to enhance the learning process from another domain. [48]
As an illustration, consider Web-document categorization, where the objective is to group a given Web content into a number of predetermined categories. For instance, university webpages that are linked to category information discovered through earlier manual labeling efforts can be considered labeled instances in the field of Web document classification. There might not be enough labeled training data for a classification problem on a newly developed website in which the data traits or distributions might differ. It might not be possible to easily transfer the webpage classifiers that were already discovered on the university website to the new website as a result. In these circumstances, it would be beneficial if we could apply our classification expertise to the new area. [38]

Transfer learning in practice

The following is the most prevalent manifestation of transfer learning within the world of deep learning:

1. Take network layers from a model that has been trained before. [25]

2. Freeze them to prevent losing any of the knowledge they carry during upcoming training sessions. [25]

3. Upon the top the previously frozen layers, add some new trainable layers. On a new dataset, they shall learn how to make predictions using the older features. [25]

4. On your dataset, train the new layers. [25]

A last, optional step is fine-tuning, which involves unfreezing the model as a whole (or a portion of it) and retraining it using the new data at an extremely low learning rate. By incrementally adjusting the already trained features based on new data, this has the potential to achieve considerable improvements. [25]

Transfer learning use cases

Below are two use cases in transfer learning that are noteworthy:

- **Fine-tuning and improving an existing model:** In this type of transfer learning, a CNN model is trained on a huge dataset like ImageNet, and the last "classifier layer" is swapped out for information particular to the fine-tuning dataset. Some variants will keep updating all of the layers through backpropagation, while others will only do so for the later layers. This is because there is less of a need to update the earlier-layer features as many of them are applicable to all kinds of visual processing. The later layers, which are more appropriate to train on the dataset specific to the domain, concentrate on merging these smaller-scale features in task-specific ways. [39]

- **Feature extraction with an already established convolutional model:** In this kind of transfer learning, the focus is on the output layer, the final fully connected layer, and use the remaining CNN layers as a "feature extractor" for the smaller dataset. Due to the fact that a pretrain dataset like ImageNet would have 1,000 separate class outputs at the output layer, thus, it might not be as useful for a task that is specialized to a domain. [39]

When to employ transfer learning?

Try applying transfer learning in any of the following circumstances (or a mixture of them):

- A small insufficient training dataset [39].

- Both base dataset and training dataset share visual features [39].

2.10.5 Inception-ResNet-v2

Inception-ResNet-v2 is pre-trained convolutional neural network. It was trained on over a million photos from the ImageNet dataset. The network has 164 layers and can identify photos into 1,000 different categories of objects, including keyboards, mice, pencils, and a variety of animals. As an outcome, the network has acquired detailed depictions of features for a diverse set of images. The network's picture input size is 299-by-299 pixels. [14]

On the output layer, the network employs the softmax activation function, while the residual blocks employ the relu activation function [45]. Inception-ResNet-v2 is the only pre-trained model that is being covered because it is the model that will be used to develop the deep learning model later on.

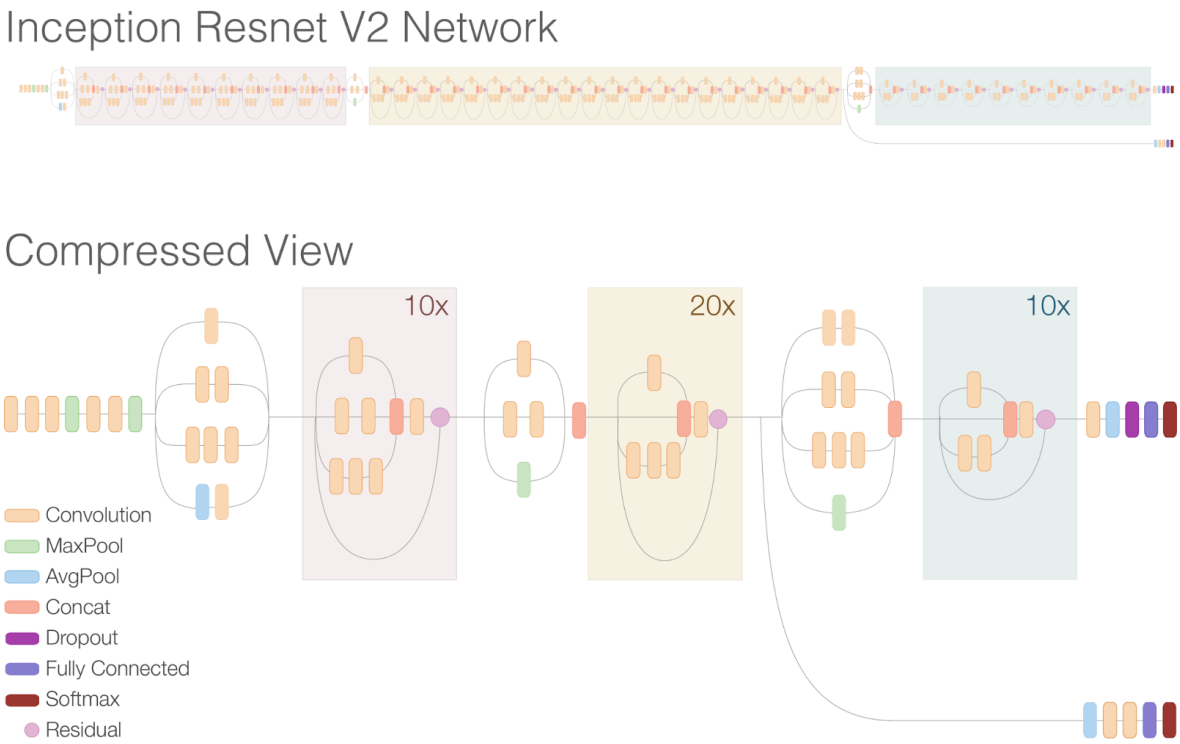


Figure 2.14: Shema for the Inception-ResNet-v2 network [23]

2.10.6 Conclusion

Even though artificial intelligence research has made great gains, there are still many obstacles to be addressed before scientists can replicate human-level reasoning and intelligence (mostly in terms of computational and processing power and other various problems).

Chapter 3

Proposed model: Implementation and results

3.1 Introduction

The general idea behind the issue that this work attempts to solve will be covered at the beginning of this chapter. There will then be a brief description of the tools (programming language, frameworks, and libraries) that were used, followed by the presentation of the end result, which will take the form of a web application.

3.2 The general idea

The first chapter provides an overview of a very significant health issue that humanity is dealing with: the exponential growth in newly diagnosed melanoma cases. In the other hand, chapter 2 discusses one of the most popular research disciplines today, artificial intelligence, as well as one of it prominent sub-fields: *deep learning*.
The goal here is to harness and apply the aforementioned fancy AI techniques (Deep Learning, Convolutional Neural Networks, ...) to assist patients who fear they may have melanoma in early detection. This is a traditional binary picture classification task. The deep learning model, which in this case is a convolutional neural network, will be trained to categorize mole photos as benign or malignant almost precisely.
The final product (see figure 3.1) is a typical web application in which the user is presented with a compact user interface (UI) where he may upload an image of the suspected mole and send it to the backend, where the trained model will predict whether the user's input is benign or malignant and return the answer back to the user.

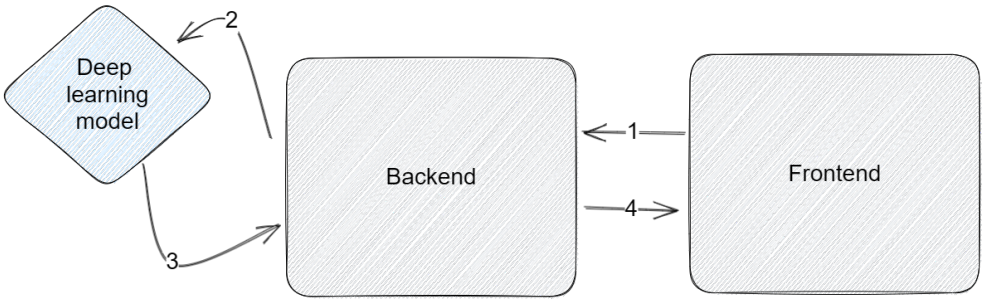


Figure 3.1: Application's high level architecture

3.3 The Python programming language

Figure 3.2: Python [20]

Python is a robust programming language that is simple to learn. Its object-oriented programming methodology is straightforward but efficient, and it includes good high-level data structures. Python is a fantastic language for scripting as well as for developing fast applications in many domains on most platforms because to its clean syntax, dynamic typing, and nature of being an interpreted language. [15]
In Python, a programmer can divide the code into modules that are capable of being used in additional Python projects. It includes a sizable number of common modules for which one might utilize as the foundation of their projects or as starter examples for learning how to write code in Python. Some of these modules include apis to graphical user interfaces (GUI) toolkits like Tk as well as file input/output, system calls, sockets, and other functions. [1]

3.4 Frameworks and libraries

3.4.1 TensorFlow

Figure 3.3: Tensorflow logo [19]

TensorFlow is a comprehensive end-to-end freely available machine learning framework that makes it simple for both novices and experts to develop machine learning models [19]. The high-level APIs of TensorFlow depends upon the Keras API standard for designing and training artificial neural networks [19].

3.4.2 Keras

Figure 3.4: Keras logo [6]

Keras is an application programming interface (API) created for humans, not machines. Keras adheres to best practices for lowering cognitive load: it provides consistent and

straightforward APIs, reduces the amount of user activities required for typical use cases, and delivers clear and actionable error signals. Keras also prioritizes creating excellent documentation and development guides. [6]
Keras is designed to provide developers who want to release apps with machine learning functionality an unfair advantage. The key features of Keras include quick debugging, elegant and succinct code, maintainability, and deployability. Your codebase will be smaller, more readable, and simpler to iterate on if you select Keras. While TF Serving, TF Lite, and TF.js make it simpler to publish them across all platforms (server, mobile, browser, embedded). [6]

3.4.3 NumPy

Figure 3.5: NumPy logo [12]

NumPy is the foundational Python library for scientific computing. It is a Python library that includes a multidimensional array object, various derived objects (such as masked arrays and matrices), and a variety of routines for performing fast array operations such as mathematical, logical, shape alteration, organizing, picking, I/O, Fourier transforms, linear algebra, statistical operations, random simulation, and much more. [12]

3.4.4 Matplotlib

Figure 3.6: Matplotlib logo [10]

Python's Matplotlib toolkit provides a complete tool for building static, animated, and dynamic visualizations. Matplotlib makes difficult things possible and simple things easy. [10]
Employing Matplotlib in a project can help with:

- Produce plots fit for publication. [10]
- Make interactive charts with zoom, pan, and update capabilities. [10]
- Change the visual style along with the layout. [10]
- Export to a variety of file formats. [10]
- Use a diverse set of third-party packages based on Matplotlib. [10]

3.4.5 FastAPI

Figure 3.7: FastaApi logo [5]

Based on common Python type hints, FastAPI is a cutting-edge, efficient (high-performance), web framework for creating restful APIs with Python 3.7+ [5]. The framework key characteristics include:

- Fast: Extremely high performance, comparable to Go and NodeJS (due to Pydantic and Starlette). [5]
- Fast to code: Approximately double or triple the rate at which features are developed. [5]
- Excellent editor support, making debugging takes less time. [5]
- Easy: Created to be simple to use and comprehend. less time reading the documentation. [5]
- Reduce the amount of duplicate code. [5]
- Robust: Get code that is ready for production. With interactive documentation that is generated automatically. [5]

3.4.6 Svelte

Figure 3.8: Svelte logo [18]

Svelte is a revolutionary new technique to creating user interfaces. Unlike popular frameworks such as React and Vue, which do the majority of their heavy lifting in the web browser, Svelte moves everything into a compile stage that occurs during the building of the project. [18]
Svelte produces code that carefully updates the Document Object Model (DOM) when the state of the application changes, rather than using approaches like virtual DOM diffing. [18]

3.5 Preparing the dataset

Only a subset of the well-known ISIC 2019 challenge dataset [46, 26, 28] was used for this work. This custom dataset will have 9522 photos separated into two groups: melanoma and non-melanoma. It contains 5000 non-melanoma photos, accounting for 52.51 percent of the dataset, and 4522 melanoma images, accounting for 47.49 percent of the overall dataset.

Class	Number	Class distribution
Melanoma	4522	47.49%
Non-melanoma	5000	52.51%

Table 3.1: An overview of the dataset

3.5.1 Resizing images

The input layer will expect a three-dimensional vector of the form (image weight x image height x number of channels), as previously stated in the CNN architecture section. As a result, it is critical that each image in the dataset have the exact same shape (Standardizing the dataset). Going through all 9522 images and manually resizing them is a time-consuming operation in and of itself. For this purpose, the Python Pillow library will be used instead.
This library supports a wide range of file formats, has an efficient inner representation, and has rather robust image processing features. The core image package was created to provide quick access to data contained in a few fundamental pixel formats. It should be a good starting point for a broad image processing program. [13]

```
import os
from PIL import Image


def resize(source, filename, dest):
  full_path = os.path.join(source, filename)
  image = Image.open(full_path)
  image = image.resize((256, 256), Image.LANCZOS)
  image.save(dest, 'JPEG', quality=95)

def run_resize(directory):
  print(f'\tresizing {directory}')
  data_path = os.path.join(os.getcwd(), f'ISIC2019/raw_dataset/{
                                    directory}')
  dataset_images = os.listdir(data_path)
  for img_name in dataset_images:
    resize(
      source=data_path,
      filename=img_name,
      dest=os.path.join(os.getcwd(),
      f'ISIC2019/resized_dataset/{directory}/{img_name.split(".")[0]
                                    }_resized.jpeg')
    )
    print(f'\033[1;32m DONE resizing the {img_name} image')
  print(f'\033[1;32m DONE resizing the {directory} images')
```

This simple script will iterate over every single image in a given directory, resize it to 256x256 px with keep 95% of its quality. After the resizing being complete the new image will be saved to a given destination folder.

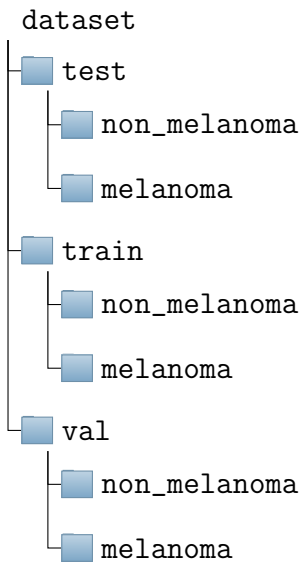
3.5.2 Splitting the dataset

When training a deep learning model, it is always a good practice to split the dataset in question into 3 parts: a **training dataset** used to train the model, a **validation dataset** used to validate the accuracy of the model during training, and a **testing dataset** used to measure the overall performance of the trained model. The split ratio favored by the industry is as follows: 80% of the dataset will be allocated for training, 10% for validation, and the remaining 10% will be used for testing.
Just like resizing the dataset images, splitting the dataset manually is a tedious consuming task. Luckily, the *split-folders* Python library comes in handy. it will randomly split the entire dataset onto 3 parts: train, validation and test. The code presented below does the job:

```
import splitfolders

def run_split(input_dir):
  output_dir = os.path.join(os.getcwd(), 'raw_dataset')
  output_dir = os.path.join(output_dir, 'final_dataset')
  splitfolders.ratio(input_dir, output=output_dir, seed=1337, ratio=
                                    (.8, .1, .1), group_prefix=None)
```

The resulting folder structure will look like this:



Dataset	Number of images
Train set	7617
Validation set	952
Test set	953

Table 3.2: Number of images in each dataset

3.6 Training the model

Now that the dataset has been processed and organized in accordance with best practices to meet the needs of our model, it is time to design and train the neural network that will be responsible for identifying the difference between malignant and benign lesions.

3.6.1 Importing libraries

The necessary libraries must be imported into the workspace in order to complete the task of developing the neural network. TensorFlow, keras, numPy, and matplotlib are those libraries.

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import numpy as np
```

3.6.2 Load the train, validation and test dataset

There are various global constants that must be defined before loading the data. After extensive testing, it turned out that the ideal batch size (the number of images loaded into memory at once) for this dataset would be 64, with the training epochs (the number of iterations through the complete training dataset) set to 200. In addition, the image size and number of channels will be assigned for easy access (these will be useful when creating the neural network input shape). Along with the directory in which the datasets are stored.

```
IMG_SIZE = 256
CHANNELS = 3
BATCH_SIZE = 64
EPOCHS = 200
DATASET_DIR = '/kaggle/working/dataset'
```

The code below, load the training, validation and testing datasets into train_ds, val_ds and test_ds variables respectively.

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  os.path.join(DATASET_DIR, 'train'),
  seed=123,
  image_size=(IMG_SIZE, IMG_SIZE),
  batch_size=BATCH_SIZE,
  shuffle=True)

val_ds = tf.keras.utils.image_dataset_from_directory(
  os.path.join(DATASET_DIR, 'val'),
  image_size=(IMG_SIZE, IMG_SIZE),
  batch_size=BATCH_SIZE)

test_ds = tf.keras.utils.image_dataset_from_directory(
  os.path.join(DATASET_DIR, 'test'),
  seed=123,
  image_size=(IMG_SIZE, IMG_SIZE),
  batch_size=BATCH_SIZE,
  shuffle=True)
```

Plotting the training dataset

Plotting the training dataset would result in the following:

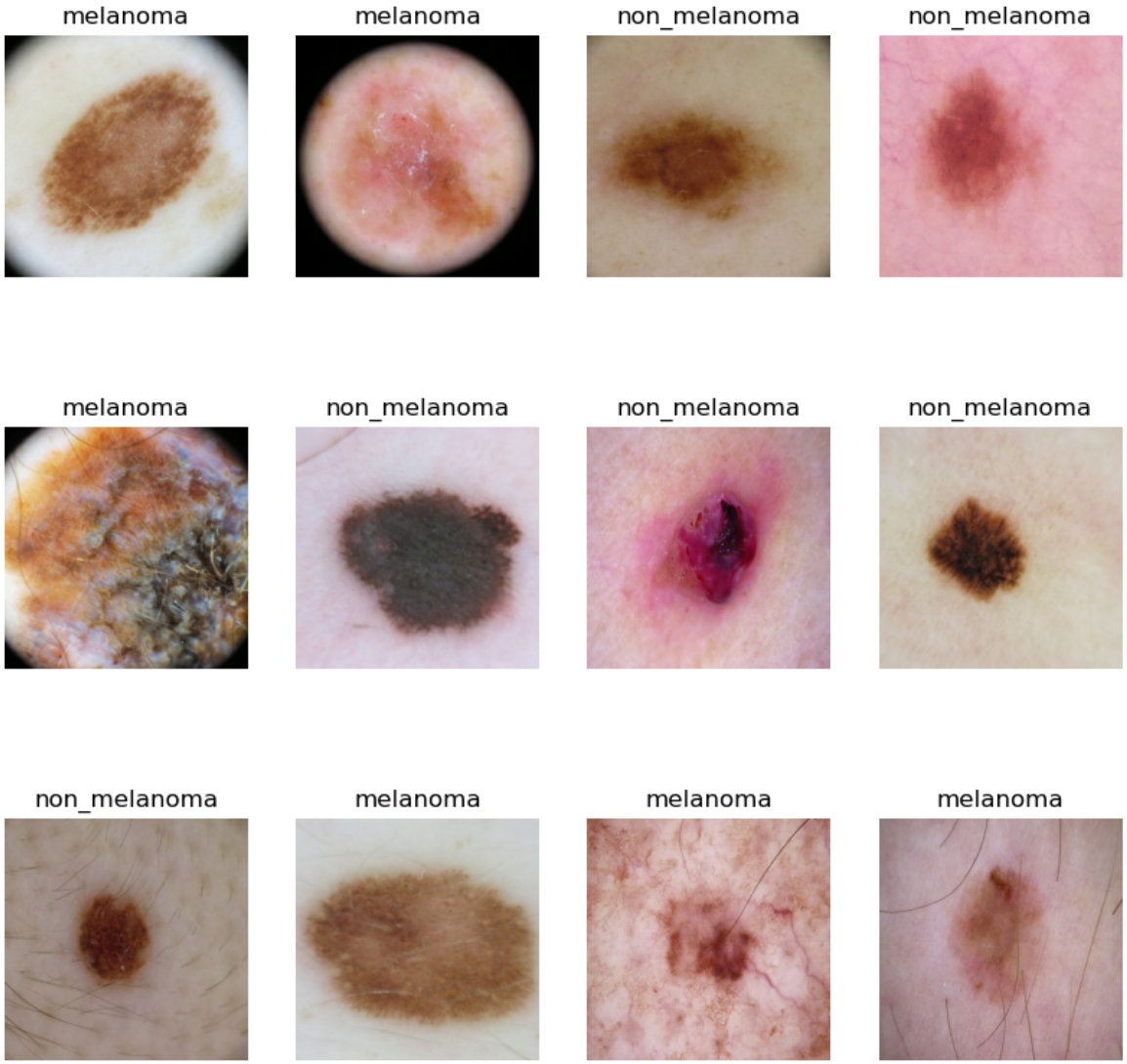


Figure 3.9: Training dataset (comes from of the ISIC2019 challenge dataset [46, 26, 28])

3.6.3 Data augmentation

Data augmentation artificially expands the training set by generating numerous realistic versions of each training case. This reduces overfitting and qualifies this as a regularization strategy. The produced images should be as close to reality as possible: ideally, a person should not be able to distinguish whether a picture contained in the augmented training data was augmented or not. [32]

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip("horizontal_and_vertical"),
  tf.keras.layers.RandomRotation(0.4),
  tf.keras.layers.RandomTranslation(
          height_factor=0.1, width_factor=0.1),
  tf.keras.layers.RandomContrast(factor=0.1),
  tf.keras.layers.RandomZoom(0.4),
```

```
])
```

Applying the *data_augmentation* layer on a given image from the training set for instance, will produce:

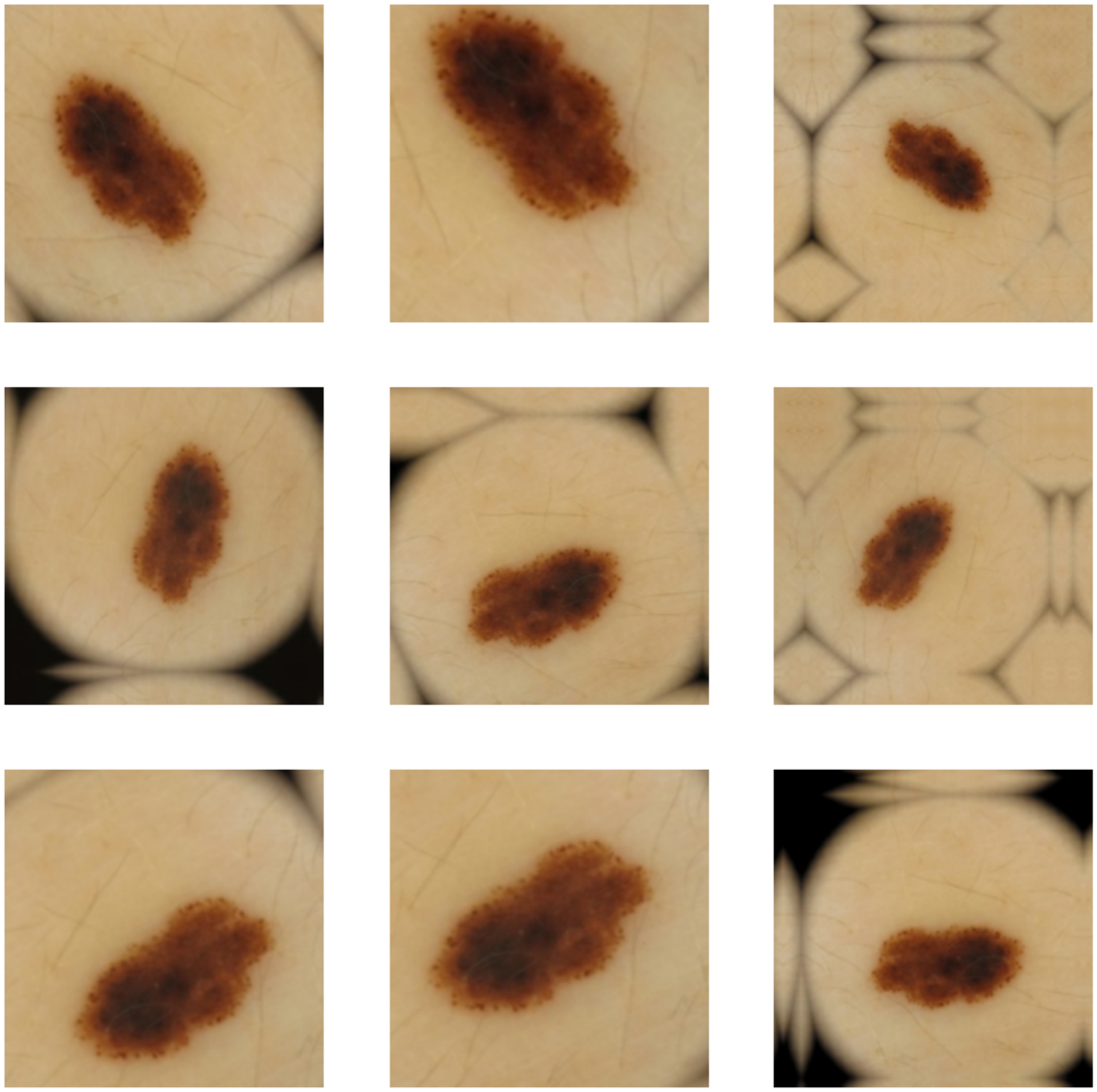


Figure 3.10: Performing data augmentation (the image comes from of the ISIC2019 challenge dataset [46, 26, 28])

3.6.4 Building the model

Transfer learning is being used to boost our model due to the tiny size of the dataset (only 9522 images). Transfer learning, as previously stated in section 2.10.4, is a technique that is typically used for assignments where your dataset contains insufficient data to train a full-scale model from scratch.
After a long series of trial and error with various pretrained models, it was determined that the Incepretion-Resnet-V2 model is the best pretrained model for this specific scenario.

Layer (type)	Output shape	Parameters
sequential (Sequential)	(None, 256, 256, 3)	0
rescaling (Rescaling)	(None, 256, 256, 3)	0
inception_resnet_v2 (Functional)	(None, 6, 6, 1536)	54336736
global_average_pooling2d	(None, 1536)	0
dense (Dense)	(None, 1024)	1573888
dense_1 (Dense)	(None, 1024)	1049600
dense_2 (Dense)	(None, 1024)	524800
dense_3 (Dense)	(None, 512)	262656
dense_4 (Dense)	(None, 512)	131328
dense_5 (Dense)	(None, 64)	65792
dense_6 (Dense)	(None, 64)	16448
dropout (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 2)	130

Table 3.3: The model architecture

The model will have a total of 57961378 parameters, of which only 3624642 are trainable and the rest are frozen. A Rescaling layer has been introduced for normalizing input values (which were originally within the [0, 255] interval) into [-1, 1] [25]. Ahead of the classification layer, a Dropout layer was added for regularization (to reduce overfitting) [25]. The training property is set to False and then passed to the base model, causing it to run in inference mode and not update the batchnorm statistics even after the basic model is unfreezed for fine-tuning [25].
The model was built using the following function:

```
def build_model():
  input_shape = (IMG_SIZE, IMG_SIZE, CHANNELS)
  n_classes = 2

  base_model = tf.keras.applications.InceptionResNetV2(
  include_top=False,
  input_shape=input_shape,
  classes=n_classes,
  weights='imagenet')

  base_model.trainable = False

  model = tf.keras.models.Sequential()
```

```
    model.add(tf.keras.Input(shape=input_shape))
    model.add(data_augmentation)
    model.add(tf.keras.layers.Rescaling(1./127.5, offset=-1))
    model.add(base_model)
    model.add(tf.keras.layers.GlobalAveragePooling2D())
    model.add(tf.keras.layers.Dense(1024, activation='relu'))
    model.add(tf.keras.layers.Dense(1024, activation='relu'))
    model.add(tf.keras.layers.Dense(512, activation='relu'))
    model.add(tf.keras.layers.Dense(512, activation='relu'))
    model.add(tf.keras.layers.Dense(256, activation='relu'))
    model.add(tf.keras.layers.Dense(256, activation='relu'))
    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dropout(0.3))
    model.add(tf.keras.layers.Dense(n_classes, activation='softmax'))

    return model
```

Training

During model training, it has been generally beneficial to gradually reduce the learning rate, the *ExponentialDecay* schedule will be used for this purpose [7]. ModelCheckpoint callback is utilized in conjunction with model.fit() training to store a model or just weights every specified interval so that the model or its weights can be restored later to continue training from the stored state [8].

```
initial_learning_rate = 1e-3
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
  initial_learning_rate, decay_steps=100000, decay_rate=0.96,
                                    staircase=True
)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(
  "classification_model.h5")
```

The *model.fit()* function is responsible for the training loop. The model training will last 200 epochs.

```
METRICS = [
  "accuracy",
]

model.compile(
  optimizer=tf.keras.optimizers.Adam(learning_rate=lr_schedule),
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=
                                    False),
  metrics=METRICS
)

history = model.fit(
  train_ds,
  epochs=EPOCHS,
  validation_data=val_ds,
  callbacks=[checkpoint_cb])
```

After training the model, it is time to measure its performance on the test set. Running the code below will yield an accuracy of **79.85%**, and **0.48** loss (started from 0.57 loss at epoch 1).

```
model.evaluate(test_ds)
```

Fine tuning

Since the model performance is not yet satisfying, the base model has to be unfreezed and also the model as a whole need to be trained from start to finish assuming a low learning rate.

```
def unfreeze_model(model):
  for layer in model.layers[-30:]:
    if not isinstance(layer, layers.BatchNormalization):
      layer.trainable = True

  optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
  model.compile(optimizer=optimizer,
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=
                                    False),
    metrics=["accuracy"])

unfreeze_model(model)
EPOCHS = 40
hist = model.fit(train_ds, epochs=EPOCHS, validation_data=val_ds)
```

After 40 epochs, when measuring the model performance on the testing set, the results will yield an accuracy of **89.19%**, and **0.37** loss (started from 0.50 loss at epoch 1).

The overall performance of the model

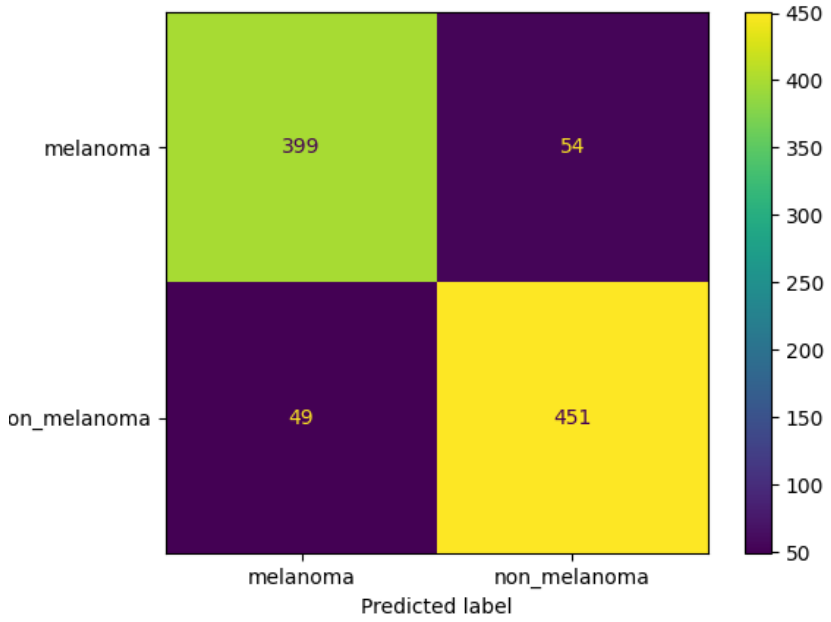


Figure 3.11: Model performance on each class

Using the scikit-learn library to measure the overall model performance, will output that the provided model is accurate **89.19%** of the times.

```
import sklearn.metrics
acc = sklearn.metrics.accuracy_score(actual, predicted)
```

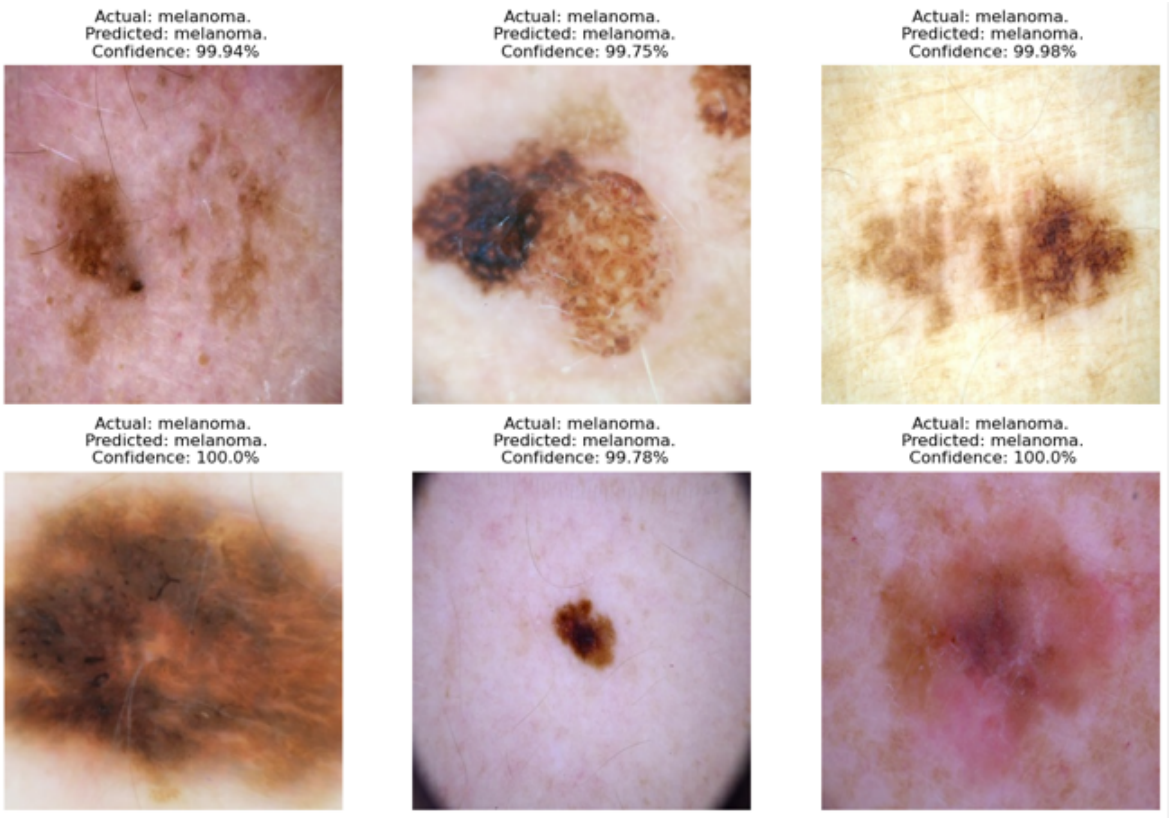


Figure 3.12: Model predictions

3.7 Building the web application

In order to make the deep learning model accessible to the end user, it will be incorporated with a simple and elegant web interface. The web application will be made up of a backend and a frontend. The backend is essentially a Restful web service with a single POST endpoint, whereas the frontend is the client that consumes that web service. (See figure 3.1). The FastAPI web framework will be used to build the backend, while Svelte will be used to build the frontend. The end result should look like figure 3.13.
In a nutshell, the backend endpoint is as follows:

```
@app.post("/predict")
async def predict(file: UploadFile = File(...)):
  image = read_file_as_image(await file.read())
  img_batch = np.expand_dims(image, 0)

  predictions = MODEL.predict(img_batch)

  predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
  confidence = round(100 * (np.max(predictions[0])), 2)
```

```
    data = {"class": predicted_class, "confidence": confidence}
    headers = {'Access-Control-Allow-Origin': 'http://localhost:5173'}
    return JSONResponse(content=data, headers=headers)
```

The logic for the frontend is implemented as follows:

```
let input, image, placeholder, result;
let showImage = false, showResult = false;

function onChange() {
  const file = input.files[0];
  if (file) {
    showImage = true;
    const reader = new FileReader();
    reader.addEventListener("load", () => image.setAttribute("src",
                                    reader.result));
    reader.readAsDataURL(file);
    return;
  }
  showImage = false;
}

async function upload() {
  const formData = new FormData();
  formData.append("file", input.files[0]);
  const upload = fetch("http://localhost:5000/predict", {
    method: "POST",
    body: formData,
  }).then((response) => response.json())
  .then((res) => {
    result = res;
    if (result.class === "non_melanoma") result.class = "begnin";
    else result.class = "malignant";
    showResult = true;
  })
  .catch((error) => console.error("Error:", error));
}
```

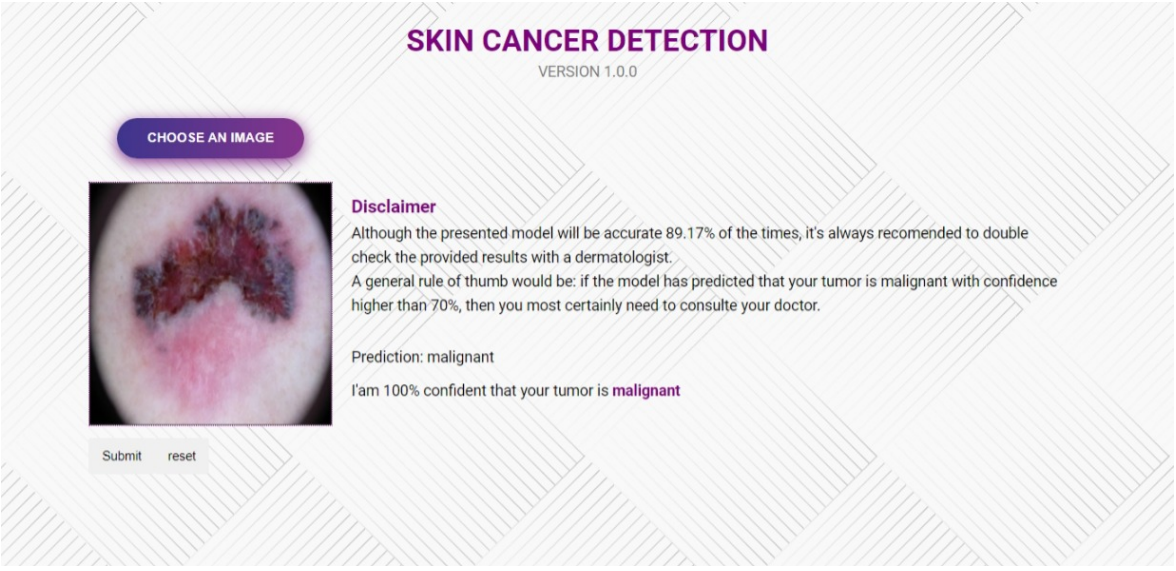


Figure 3.13: The web application

3.8 Conclusion

Every deep learning model requires a massive quantity of data to effectively learn from. This is not the situation in the real world, particularly in medical fields, where high-quality labeled data is scarce. For example, when it came to classifying melanoma, the most difficult obstacle was the tiny dataset. Fortunately, such a challenge was overcome with the help of transfer learning.

General conclusion

Through out the course of this work, we were able to deliver an end-to-end AI powered application that aims to help regular people clear their doubts on whether the suspected lesion is a melanoma or not. In melanoma like in any other type of cancer, early detection is crucial, and that is what the proposed model tries to help with.

Tracking back, the biggest pain point was the insufficient amount of data. To build an accurate performant model, 9522 images were simply not enough even with the use of fancy techniques like transfer learning. With that in mind, since freely accessible data is not enough, a great course of action would be to gather more data from proprietary resources such as private hospitals, clinics and so forth. Another suitable enhancement would be to introduce *online learning* to the model, this will allow to learn from new data and update its weights while still running on production.

Bibliography

[1] 1. whetting your appetite. (n.d.). python documentation. https://docs.python.org/3/tutorial/appetite.html. Accessed on May 28, 2023.

[2] Alan turing — biography, facts, computer, machine, education, & death — britannica. (2023, april 24). https://www.britannica.com/biography/Alan-Turing. Accessed on May 5, 2023.

[3] Anatomy of the skin-stanford children health. https://www.stanfordchildrens.org/en/topic/default?id=anatomy-of-the-skin-85-P01336. Accessed on April 19, 2023.

[4] Basal cell carcinoma—symptoms and causes. (n.d.). mayo clinic. https://www.mayoclinic.org/diseases-conditions/basal-cell-carcinoma/symptoms-causes/syc-20354187. Accessed on April 19, 2023.

[5] Fastapi. (n.d.). https://fastapi.tiangolo.com/. Accessed on May 28, 2023.

[6] Keras: Deep learning for humans. (n.d.). https://keras.io/. Accessed on May 28, 2023.

[7] Keras documentation: Exponentialdecay. https://keras.io/api/optimizers/learning_rate_schedules/exponential_decay/. Accessed on June 2, 2023.

[8] Keras documentation: Modelcheckpoint. https://keras.io/api/callbacks/model_checkpoint/. Accessed on June 2, 2023.

[9] Les registres du cancer en afrique. http://www.santetropicale.com/santemag/algerie/poivue61.htm. Accessed on April 19, 2023.

[10] Matplotlib—visualization with python. (n.d.). https://matplotlib.org/. Accessed on May 28, 2023.

[11] Melanoma of the skin—cancer stat facts. (n.d.). seer. https://seer.cancer.gov/statfacts/html/melan.html. Accessed on April 23, 2023.

[12] Numpy documentation—numpy v1.24 manual. (n.d.). https://numpy.org/doc/stable/. Accessed on May 29, 2023.

[13] Pillow. (n.d.). pillow (pil fork). https://pillow.readthedocs.io/en/stable/index.html. Accessed on May 31, 2023.

[14] Pretrained inception-resnet-v2 convolutional neural network—matlab inceptionresnetv2. (n.d.). https://www.mathworks.com/help/deeplearning/ref/inceptionresnetv2.html. Accessed on June 2, 2023.

[15] The python tutorial. (n.d.). python documentation. `https://docs.python.org/3/tutorial/index.html`. Accessed on May 28, 2023.

[16] Skin cancer, anatomy, thanc guide. (2019, april 10). `https://thancguide.org/cancer-types/skin/anatomy/`. Accessed on April 18, 2023.

[17] Squamous cell carcinoma of the skin—symptoms and causes. (n.d.). mayo clinic. `https://www.mayoclinic.org/diseases-conditions/squamous-cell-carcinoma/symptoms-causes/syc-20352480/`. Accessed on April 19, 2023.

[18] Svelte • cybernetically enhanced web apps. (n.d.). `https://svelte.dev/`. Accessed on May 28, 2023.

[19] Tensorflow core — machine learning for beginners and experts. (n.d.). `https://www.tensorflow.org/overview/`. Accessed on May 29, 2023.

[20] Welcome to python.org. (2023, may 22). python.org. `https://www.python.org/`. Accessed on May 28, 2023.

[21] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, 2012.

[22] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10(978):3, 2018.

[23] Alex Alemi. Improving inception and image classification in tensorflow. `https://ai.googleblog.com/2016/08/improving-inception-and-image.html`, August 2016. Accessed on June 2, 2023.

[24] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[25] François Chollet. Keras documentation: Transfer learning & fine-tuning. `https://keras.io/guides/transfer_learning/`, April 2020. Accessed on June 2, 2023.

[26] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.

[27] William P Coleman, Philip R Loria, Richard J Reed, and Edward T Krementz. Acral lentiginous melanoma. *Archives of dermatology*, 116(7):773–776, 1980.

[28] Marc Combalia, Noel CF Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Cristina Carrera, Alicia Barreiro, Allan C Halpern, Susana Puig, et al. Bcn20000: Dermoscopic lesions in the wild. *arXiv preprint arXiv:1908.02288*, 2019.

[29] Thomas L Diepgen and V Mahler. The epidemiology of skin cancer. *British Journal of Dermatology*, 146(s61):1–6, 2002.

[30] Friederike Erdmann, Joannie Lortet-Tieulent, Joachim Schüz, Hajo Zeeb, Rüdiger Greinert, Eckhard W Breitbart, and Freddie Bray. International trends in the incidence of malignant melanoma 1953–2008—are recent generations at higher or lower risk? *International journal of cancer*, 132(2):385–400, 2013.

[31] Robert J Friedman, DarrellS Rigel, Mark K Silverman, Alfred W Kopf, and Katrien A Vossaert. Malignant melanoma in the 1990s: The continued importance of earlydetection and the role of physician examination and self-examination of theskin. *CA: A Cancer Journal for Clinicians*, 41(4):201–226, 1991.

[32] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[33] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[34] Ali Hendi and Juan-Carlos Martinez. *Atlas of skin cancers: practical guide to diagnosis and treatment*. Springer, 2011.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[36] William D Losquadro. Anatomy of the skin and the pathogenesis of nonmelanoma skin cancer. *Facial Plastic Surgery Clinics*, 25(1):283–289, 2017.

[37] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[38] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[39] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[40] Darrell S Rigel and John A Carucci. Malignant melanoma: prevention, early detection, and treatment in the 21st century. *CA: a cancer journal for clinicians*, 50(4):215–236, 2000.

[41] Andrej A Romanovsky. Skin temperature: its role in thermoregulation. *Acta physiologica*, 210(3):498–507, 2014.

[42] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[43] I. Salian. (2018, august 2). nvidia blog: Supervised vs. unsupervised learning. nvidia blog. `https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/`. Accessed on May 9, 2023.

[44] Hyuna Sung, Jacques Ferlay, Rebecca L Siegel, Mathieu Laversanne, Isabelle Soerjomataram, Ahmedin Jemal, and Freddie Bray. Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: a cancer journal for clinicians*, 71(3):209–249, 2021.

[45] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[46] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

[47] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[48] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[49] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.