MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF MOHAMED El-BACHIR El-IBRAHIMI

BORDJ BOU-ARRERIDJ

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTEMENT OF COMPUTER SCIENCE

THESIS OF END OF STUDY

A thesis submitted in partial fulfillment of the requirements for the Degree of Master in Business Intelligence Engineering

Theme

Skin Lesion Analysis Towards Melanoma Detection With Deep Convolutional Neural Network

By :

- Guissous Alla Eddine

Advisor :

Mr. Nouioua Farid

Publicly supported on 10 / 07 / 19 In front of the jury composed of:

President :	Mr. Bouziane Abderraouf	Lecturer A, university of BBArréridj
Rapporteur :	Mr. Nouioua Farid	Lecturer A, university of BBArréridj
Examiner :	Mr. Belazzoug Mouhoub	Assistant Professor A, university of BBArréridj
Examiner :	Mr. Maaza Sofiane	Lecturer B, university of BBArréridj
Guest :	Mr. Maach Salah	Assistant Professor A, university of BBArréridj

Academic year : 2018-2019

ملخص

الميلانوما هي نوع من سرطان الجلد الذي يعتبر الأكثر خطورة وعدوانية, يتم تشخيصها بشكل أساسي بصريًا, دقة خبراء الجلد في التشخيص ضعيفة نوعا ما إذا أخذنا في عين الإعتبار معدل الوفيات الذي يتراوح بين 80 إلى 85 بالمئة في أغلب الحالات, نسبة الشفاء مرتفعة للغاية 92٪ ، إذا تم اكتشاف سرطان الجلد مبكراً ، لذا فإن وجود طريقة آلية للكشف عن المرض وتصنيفه في وقت مبكر يعد أمرًا ضروريًا للغاية لإدارة علاج فعال وزيادة فرص البقاء على قيد الحياة، تعتبر إحدى أفضل الطرق لأتمتة المهمة هي باستخدام الشبكات العصبية التلافيفية العميقة لقدرتها على القيام بمهام عامة ومتغيرة عبر العديد من الفئات الدقيقة ويمكن أن تؤدي إلى نتائج دقيقة للغاية ، تم تصميم الحل المقترح حول بنية عدة نماذج مشهورة (VGG-Net, MobileNet, DenseNet, Inception_v3)بإستخدام نهج نقل التعلم, في الأخير يجري هذا العمل تقييما مقارنا لنهج التصنيف باستخدام نقل التعلم مقابل نموذج تم بنائه من الصفر من أجل تقييم أي منها يحقق نتائج تصنيف أفضل

Abstract

Melanoma is an dangerous and aggressive type of skin cancer, is primarily diagnosed visually, the accuracy of dermatologists is very low considering the death rate is between 80% to 85% in many cases, dermatologists must perform a dermoscopy analysis (a laboratory medical procedure) to determine if a tumor is malignant or benign, the curability is extremely high 92%, if the skin cancer is detected early, so having an automated method to detect and classification the disease earlier is very needed for administering effective treatment and increasing survival chances, one of the best method to automate the task was using deep convolutional neural Networks (CNN) for its ability to general and very variable tasks across many flour-grained object categories and can produce highly accurate results, the proposed solution is built around the VGG-Net, MobileNet, DenseNet, Inception_v3 ConvNet architecture and uses the transfer learning approach. Finally, this work performs a comparative evaluation of classification approaches using transfer learning against build from scratch architecture in order to assess which of them achieves better classification results.

Résumé

Le mélanome est un très dangereux et agressif type de cancer de la peau, est principalement diagnostiqué visuellement, commençant par un dépistage clinique initial qui est potentiellement suivi par une analyse dermoscopique, la curabilité est extrêmement élevée 92%, Si le cancer de la peau est détecté tôt, la précision des dermatologues est très faible, compte tenu du risque associé à la maladie mortelle si elle n'est pas détectée plus tôt, le taux de mortalité se situe entre 80% et 85%, il est donc indispensable de disposer d'une méthode automatisée pour détecter et classer la maladie plus tôt. l'une des meilleures méthodes pour automatiser la tâche consistait à utiliser des profonde réseaux neurones convolution, le CNN montre une capacité pour des tâches générales et très variables. à travers de nombreux catégories d'objet délicats et peut produire des résultats très précis, la solution proposée est construite autour de la VGG-Net, MobileNet, DenseNet ConvNet architecture et utilise l'approche « transfer learning ». Enfin, ce travail effectue une évaluation comparative des approches de classification en utilisant «transfer learning» contre une architecture construire à partir de zéro, afin d'évaluer lequel d'entre eux obtient les meilleurs résultats de classification.

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

Motivation

Convolution Neural Networks (ConvNets) are specialized Neural Networks, which are well suited for the processing of images. The ConvNets are not a new idea, for example in 1989 have LeCun et al. [20] a ConvNet used to recognize handwritten digits in mails, but the hype around ConvNets has started 2012 after the incredible winning of Krizhevsky et al. [21] on the ImageNet challenge [22] for classification of images into 1,000 categories. The ConvNet solution of Krizhevsky et al. had a top-5 error-rate of 15.3% compared to the second place with 26.17%. Since this, the error rate is further decreased with other ConvNet designs and further research on the training of Neural Networks. ConvNets have not only reached good results on ImageNet. They are also successfully applied on different tasks on images, like the recognition of numbers in street view images [23] or in traffic sign classification [24]. But also there are ConvNets for non-image tasks. For example, ConvNets can be also applied in natural language processing [25].

ConvNets or rather Neural Networks (NN) are very interesting machine learning methods. They have a layered structure, and every layer could learn an abstract representation of the input. This ability is called Feature Learning or known as Representation Learning [26]. Feature Learning is a method, that allows a machine to learn from raw data. The machine takes the raw data as input and learns automatically the needed features to solve the machine learning problem. For example, in image classification is the raw data an image, which is represented by an array of pixels. These arrays are fed to the ConvNet, and the ConvNet learns useful features from these images to solve the machine learning problem. In the first layer, the ConvNet could be learn to detect edges, in the next layer the arrangements of the edges and so on [25]. More generally formulated, in the lower layers are basic concepts learned, and with every further layer, die concepts are combined to higher concepts.

In traditional image classification “Machine Learning”, there must be extracted useful features from the image, which are then used on machine learning algorithms like an SVM [27]. These handcrafted features must be carefully chosen, otherwise, the classification has a bad performance.

This problem of carefully chosen features is not present in NN, but they have other difficulties. One of the difficulties is the right choice of hyperparameters and of the architecture. These have direct influence on the performance.

1.1 Background of the Problem

Skin is the largest organ of the body. Its importance comes from the way it protects the internal body tissues from the external environment, i.e., skin keeps the body temperature at a constant level, protects our body from undesirable sun radiation such as ultraviolet (UV) light exposure, prevents infections and allows the production of vitamin D, essential for many body functions [1].

The skin has the ability to reduce the harmful effects of ultraviolet (UV) radiation due to the pigment melanin that absorbs UV radiation, thus protecting the cell's nuclei from DNA damage.

It helps regulate body temperature through changes in blood flow in the cutaneous vascular system and evaporation of sweat from the surface. The skin also synthesizes vitamin D3 and permits the sensations of touch, heat, and cold. Skin has three layers called epidermis, dermis and hypodermis. The epidermis is the outermost layer of skin, provides a waterproof barrier and creates our skin tone. The dermis, located beneath the epidermis, contains tough connective tissue, hair follicles, and sweat glands. The deeper subcutaneous tissue is the hypoderm which is sometimes called fat layer. It is made of fat and connective tissue. The skin's color is created by special cells called melanocytes, which are located in the epidermis and produce the pigment melanin [2, 3]. Classification of the skin based on its reaction to UV radiation is shown in (Table 1.1) below.

Type	Definition	Description	Color
I	Always burns but never tans	Pale skin, red hair, freckles	
II	Usually burns, sometimes tans	Fair skin	
III	May burn, usually tans	Darker skin	
IV	Rarely burns, always tans	Mediterranean	
V	Moderate constitutional pigmentation	Latin American, Middle Eastern	
VI	Marked constitutional pigmentation	Black	

Table 1. 1: Classification of the skin based on its reaction to ultraviolet radiation [6]

In the past few years the number of skin cancer cases has been going up and studies announce that the frequency of melanoma doubles every 20 years [4]. Skin cancer, the most predominant type of cancer, is produced when skin cells begin to grow out of control. There are 3 main types of skin cancers: basal cell skin cancers (basal cell carcinomas), squamous cell skin cancers (squamous cell carcinomas) and melanomas, and other non-common types Generally, skin cancers that are not melanomas are commonly grouped as non-melanoma skin cancers. [5]

Checking your skin for suspicious changes (Figure 1.1), can help detect skin cancer at its earliest stages. Early detection of skin cancer gives you the greatest chance for successful skin cancer treatment.

This project is focused on detection and classification of many types of skin lesion together with melanoma, which is a fatal form of skin cancer often undiagnosed or misdiagnosed as a benign skin lesion (Figure 1.2). Melanoma can be detected by a simple visual examination since it occurs on the skin surface, but an early detection is imperative, the lives of melanoma patients depend on accurate and early diagnosis.

This poses additional challenges to the task of distinguishing among skin lesions, especially between benign or malignant tumors, due to the large imbalance in the number of samples of each class of tumors.

In recent years, high numbers of new skin cancer diagnoses, especially basal cell carcinoma and squamous cell carcinoma, The cause of the spread of this type of cancer is lack of awareness and laziness to diagnose any suspicious lesion in the skin by a specialist. Also, going to a dermatologist, can be very expensive. For the reasons described above, a public, reliable and accurate method is essential to help in early detection of skin cancer, or at least to motivate the patients to make a professional diagnosis.

Figure 1. 1 : Human back full of moles

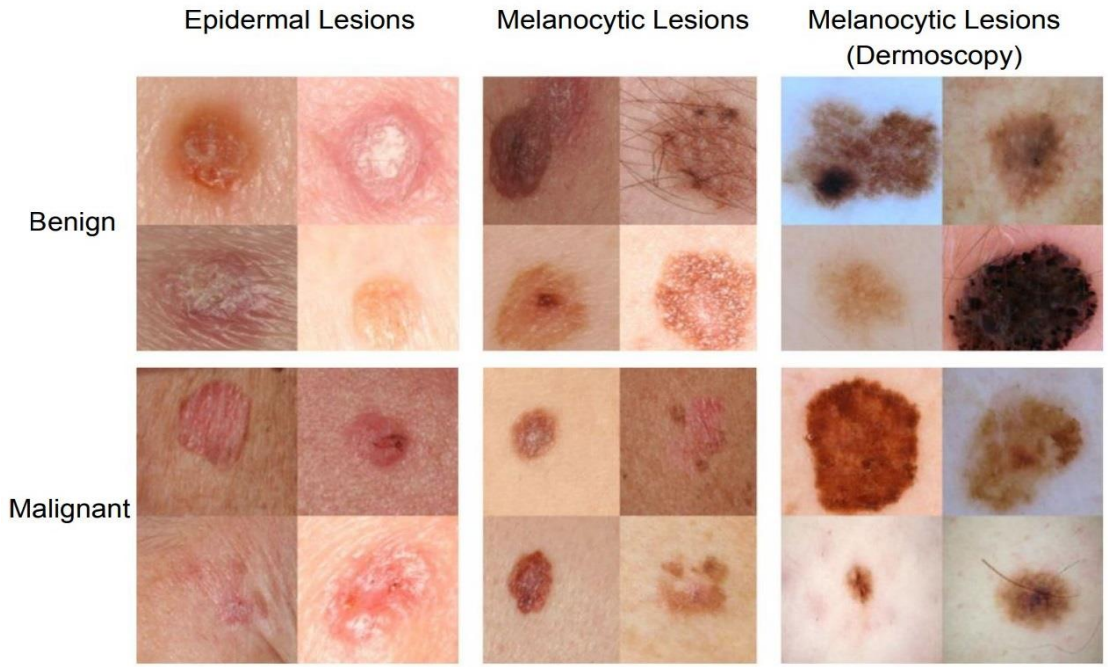


Figure 1.2 : Benign and Malignant skin lesions from the ISIC Archive dataset[7]

1.2 Statement of Purpose

The emergence of a machine learning paradigm known as deep learning and recent advances in computational power have enabled the development of intelligent medical image analysis systems that can display remarkable performance in comparison to hand-crafted features. Artificial neural networks have produced promising results when classifying skin lesions [8][9][10].

The purpose of this work is to assess the performance of Convolutional Neural Networks (also known as ConvNets or CNNs) [11] in building models for the diagnosis of skin lesions, and the production of the model for the public use. To achieve this goal, this work is divided into three parts:

- The design of a method for automatic classification of skin lesions from dermoscopic images with transfer learning.
- Comparative evaluation of classification approaches using transfer learning against build from scratch architecture.
- Production of the model as a GUI web app to facilitate the use of the model by ordinary users.

The hypothesis of this work

hypothesis 1: Generate synthetic samples by dividing original images then combine the resulting slices after being shuffled, can be as new effected way to balance the data set. However, this approach may lose information that could be contextually relevant to the CNN.

hypothesis 2 Dividing the multi-class problem to binary-class classification then combine the models responsible for each to create a multi-class model where the new model treat each class equal.

1.3 Methods and Procedures

First and foremost, all input images had to be preprocessed to be compatible with the format and size expected by the input layer of the proposed architecture.

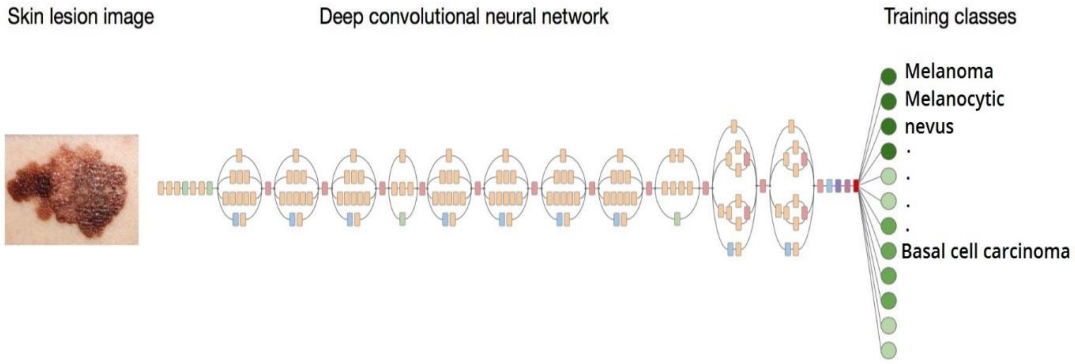


Figure 1.3 : Classification architecture with Deep Convolutional Neural Network

1.3.1 Skin Lesion Classification

The image classification task was addressed with a pretrained ConvNet to perform a multi-class classification as either one of the eight (8) skin lesion (Figure 1.3).

The main goal of the project was comparing the transfer learning approaches and the build from scratch architecture and assess their relative benefits and limitations. All the subtasks were compared by fixing all the rest of condition and parameters of the experiments.

1.4 Work Plan

This project followed the work packages detailed in this section, with the exception of some minor modifications and deviations described in Section 1.5.3.

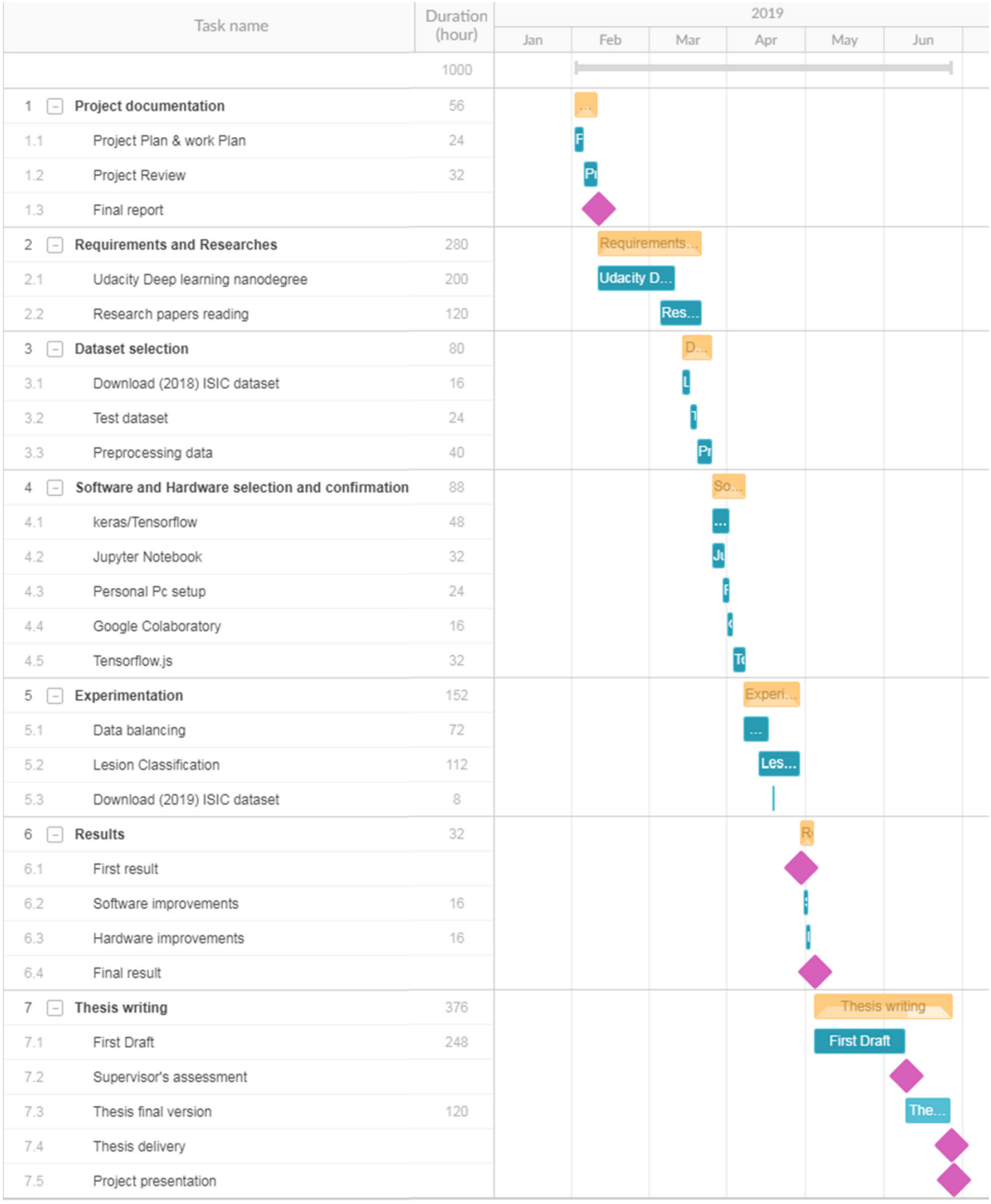
1.4.1 Work Packages

- WP 1: Project documentation
- WP 2: Requirements and Researches
- WP 3: Dataset selection
- WP 4: Software and Hardware selection and confirmation
- WP 5: Experimentation
- WP 6: Results
- WP 7: Thesis writing

1.4.2 Gantt diagram

The Gantt diagram of Figure 1.4 illustrates the work breakdown structure of the project in monthly period, a weekly period diagram is attached to the document .

Figure 1.4 : Gantt diagram month period



1.4.3 Deviations and Incidents

The only incidence the project suffered from was related to the GPU power supply. Despite the unavailability of GPUs, the documentation was insufficient, which caused delays that could have potentially jeopardize the entire project schedule. The problem was solved by using the computation service from google Colaboratory.

1.5 Significance of the study

The incidence of melanoma skin cancer has been increasing through time and it became a reason for the death of many people globally. The usual clinical practice of melanoma diagnosis is a visual inspection by the dermatologist and then taking a biopsy invasively. However, this diagnostic technique lacks visualization of morphological features which are not discernible by examination with the naked eye. In addition, the high cost of examinations and the lack of specialists prevent many patients from receiving effective treatment. Because of these challenges, developing a convolutional neural network (CNNS) model for skin lesions classification and produce it as a web application for a normal users is very important to help in early detection of skin cancer, or at least to motivate the patients to make a professional diagnosis.

1.6 Organization of the thesis

The rest of the thesis is organized as follows: Chapter 2 discusses the general structure of the skin and existing clinical melanoma diagnostic methods. It presents some of the common benign as well as malignant melanocytic and non-melanocytic skin lesions. Chapter 3 discusses the basic concepts of Machine learning, Neural Networks and Convolutional Neural Networks (CNNs) models and the general detection scheme of most CNNs models. like data preprocessing, feature extraction and classification of color images.

Chapter 4 presents the proposed skin lesion classification models and the steps followed like data preprocessing. Also discusses the data sets that have been used in the current study and the results obtained in all stages of the proposed method.

Chapter 5, is about the conclusion and future works of the study.

Chapter 2

2 Skin Biology and the Skin Lesion

Introduction

This chapter will discuss the structure and functions of the human skin. It takes into account the understanding of the skin lesions types and subtypes, also it cover the stages of the skin cancer development and the way dermatologist can diagnosis the skin lesions.

2.1 Skin Biology

2.1.1 General structure of the human skin

The human skin is divided into three main layers: the epidermis, dermis and hypodermis. (Figure 2.1) shows the anatomy of the human skin.

Epidermis

Epidermis is the outermost layer of the skin. This layer consists of many special skin cells including keratinocytes and melanocytes. Keratinocytes are cells that make a special fat which gives skin its waterproof properties. The actual skin color of humans is affected by many substances, however, the single most important substance determining human skin color is the pigment melanin, which is produced within the skin in cells called melanocytes. It determines the color of darker-skinned humans. The color of people with light skin is determined mainly by the bluish-white connective tissue under the dermis and by the hemoglobin circulating in the veins of the dermis. This layer is continuously shed and replaced every 15–30 days. The epidermis is subdivided into 5 layers. Stratum corneum is the outermost layer of the epidermis and it prevents invasion from foreign things, such as bugs and bacteria. Stratum lucidum is part of the layer which contains several clear and flat dead cells. It is a tough layer and is found in thickened skin, including the palms of the hand and soles of the feet. Stratum granulosum is composed of 3 to 4 layers of cells. Here, keratin is formed, which is a colorless protein important for skin strength. Stratum spinosum is another layer which contains cells that change shape from columnar to polygonal. Keratin is also produced here. Stratum basale is the deepest layer of the epidermis, in which many cells are active and dividing. The stratum basale is separated from the dermis layer by a basement membrane, which is a layer made of collagen and proteins [28, 29].

Dermis

The dermis is the second major layer of the skin. It is a thick layer made up of strong connective tissues. It is further divided into two levels; the upper is made of loose connective tissue, called the papillary layer, and the lower layer is made of tissue that is more closely packed, called the reticular layer. The dermis is made up of a matrix of collagen, elastin and network of capillaries and nerves. The collagen gives the skin its strength, the elastin maintains its elasticity and the capillary network supplies nutrients to the different layers of the skin. The dermis also contains a number of specialized cells and structures. These includes: hair follicles, sweat glands, sebaceous glands (produce sebum which helps lubricate skin & hair) and nails. It also plays a great role in controlling our skin temperature and acts as a cushion against mechanical injury. When injured the dermis heals through the formation of granulation tissue (a tissue rich in new blood vessels and many different cells). This tissue helps pull the edges of a cut or wound back together. It takes our body from 3 days to 3 weeks to form this tissue [29].

Hypodermis

The hypodermis contains 50% of our body's fat that helps insulate the body from heat and cold, provides protective padding, and serves as an energy storage area [28, 29].

Melanocytes

Melanocytes are pigment cells that are found in the basal layer of the epidermis and produce a protein called melanin, a brown pigment that is responsible for the skin coloration and protection against the harmful effects of ultraviolet (UV) light by absorbing some potentially dangerous UV radiation. It also contains DNA repair enzymes that help reverse UV damage, but in some cases a person which lacks the genes for these enzymes suffer high rates of skin cancer. Melanocytes can transfer melanin to other skin cells like keratinocytes through dendrites when stimulated by exposure to UV radiation and these cells help to strengthen the hair, nails, and the skin itself. When the skin is exposed to the sun, melanin production increases, which produces a tan and this is the body's natural defense mechanism against sunburn. However melanocytes do not always function as they should. For example there is a hereditary skin condition where melanocytes do not produce melanin and this results in the formation of white and oval shaped patches of skin, that gradually grow larger. In general, the human body has the same amount of melanocytes but differ in the amount of melanin produced by melanocytes [30].

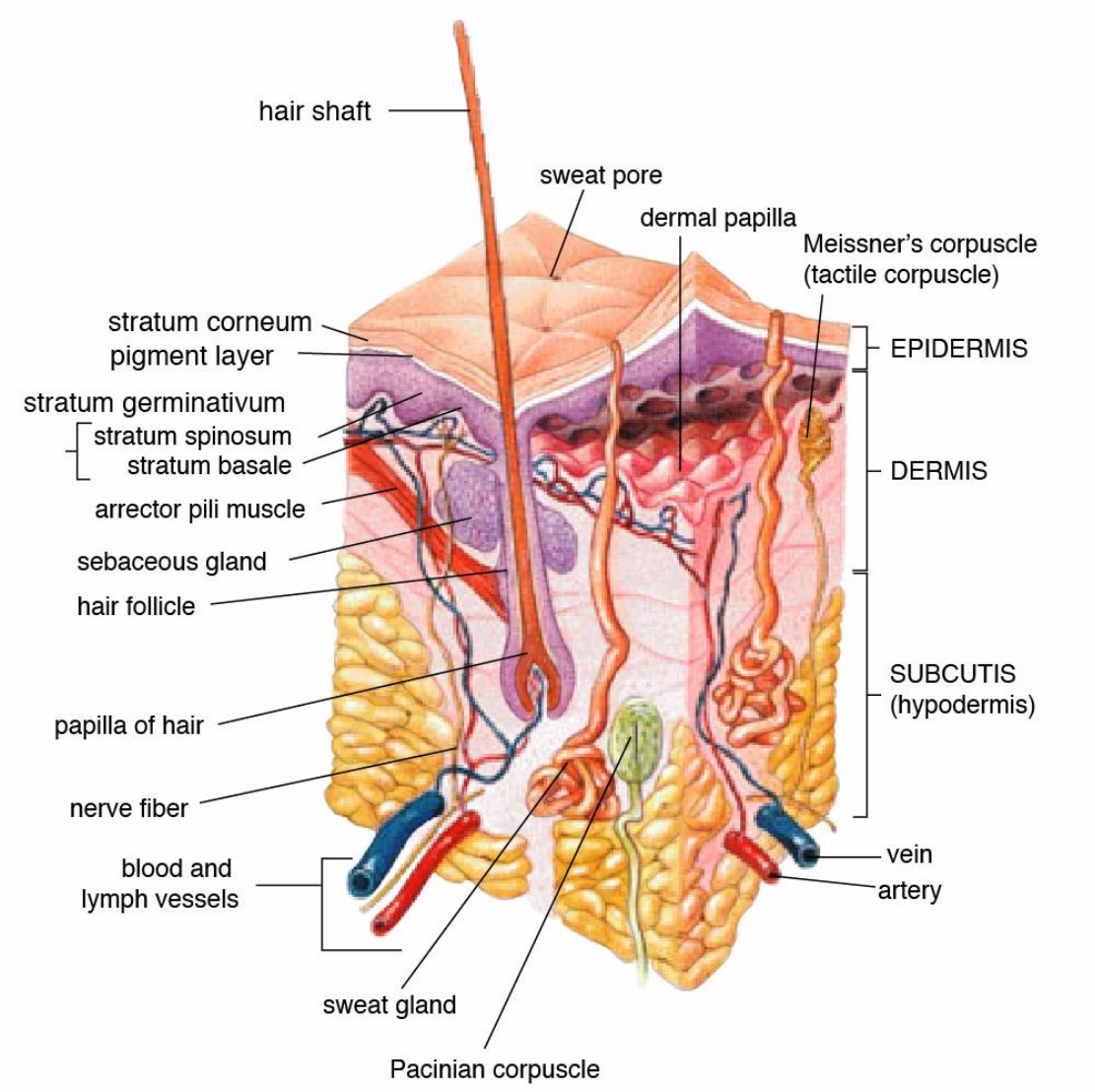


Figure 2.1: Anatomy of the skin (Courtesy of Matthew Hoffman, [31])

2.2 The Skin Lesion

2.2.1 Benign skin lesion

Benign keratosis (bkl): Also known as Seborrheic keratoses one of the most common noncancerous skin growths in older adults. usually appears as a brown, black or light tan growth on the face, chest, shoulders or back(Figure 2.2). The growth has a waxy, scaly, slightly elevated appearance. Seborrheic keratoses don't become cancerous and aren't thought to be related to sun exposure, but they can look like skin cancer [32].

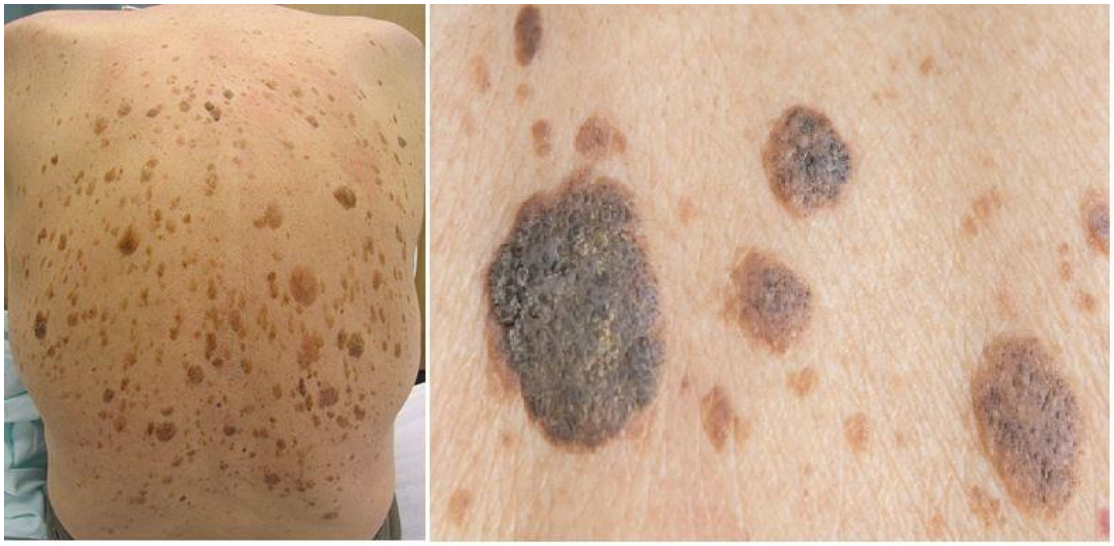


Figure 2.2: Seborrheic keratoses on the back and Close-up view

Dermatofibroma (df): Dermatofibromas are small, noncancerous (benign) skin growths that can develop anywhere on the body but most often appear on the lower legs, upper arms or upper back. These nodules are common in adults but are rare in children. They can be pink, gray, red or brown in color and may change color over the years. They are firm and often feel like a stone under the skin(Figure 2.3) [33].

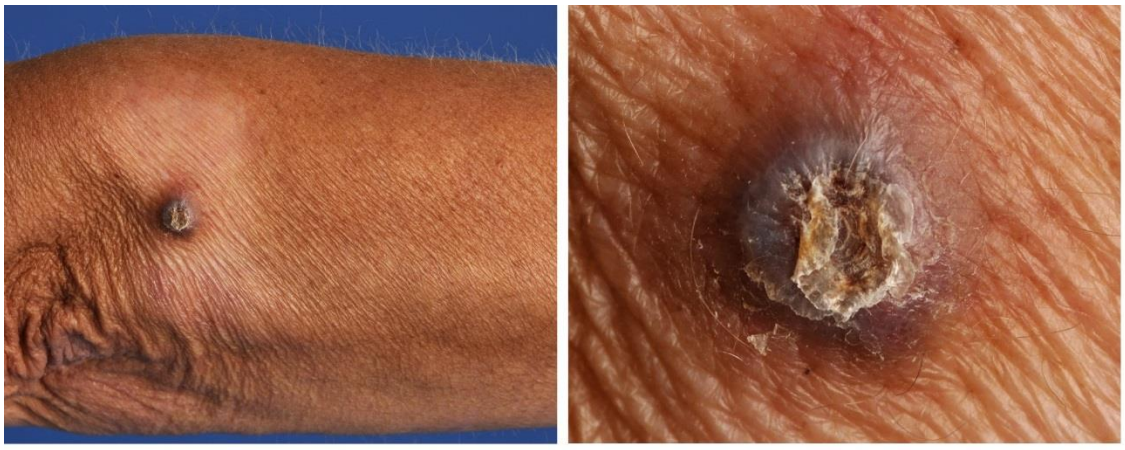


Figure 2.3: Dermatofibromas on the arm and Close-up view

Melanocytic Nevi (nv): Melanocytic nevi are pigmented moles. The word 'melanocytic' means that they are made up of the cells (melanocytes) which produce the dark pigment (melanin) that gives the skin its color. Melanocytes clustered together form nevi. This type of moles vary in color in different skin tones and they are easier to see on pink skins. Some moles are present at birth or appear within first two years of life are known as congenital melanocytic nevi. Most develop during childhood and early adult life and are consequently called acquired melanocytic nevi. The number of moles increase up to the age of 30-40. Thereafter, the number of nevi tend to decrease. New moles appearing in adulthood need to be monitored and checked if growing or changing. Moles can be found anywhere on the skin, including on the hands and feet, genitals, eyes and scalp[34]. In childhood, most moles are flat and usually circular. Later in life some become raised and more hairy (Figure 2.4), and moles on the face often become pale over time [35].

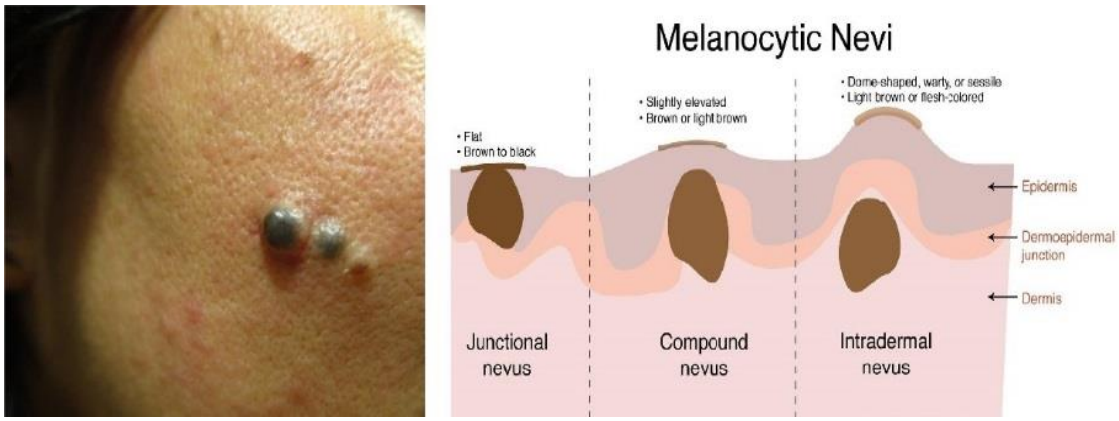


Figure 2.4: Melanocytic Nevi Development

Vascular Lesions (vasc): Vascular lesions are relatively common abnormalities of the skin and underlying tissues, more commonly known as birthmarks, generally categorized by a red or purple color and solid (Figure 2.5), well-circumscribed structures known as red clouds or lacunas [36].

Figure 2.5: Different shapes in different location for Vascular Lesions

2.2.2 Malignant skin lesion

Actinic keratoses (AK): An actinic keratosis, also known as a solar keratosis, is a crusty, scaly growth caused by damage from exposure to ultraviolet (UV) radiation. It is a common non-invasive variants of squamous cell carcinomas. They are sometimes seen as precursors that may progress to invasive squamous cell carcinoma[37] (Figure 2.6) .

Figure 2.6: Actinic keratoses is widespread in the hand

Basal cell carcinoma (bcc): Basal cell carcinoma is a common variant of epithelial skin cancer that rarely metastasizes but grows destructively if untreated (Figure 2.7 - A). It appears in different morphologic variants (Figure 2.7 B-C) (flat, nodular, pigmented, cystic, etc.)[38]. Basal cell carcinoma often appears as a slightly transparent bump on the skin (Figure 2.7 B), though it can take other forms. Most basal cell carcinomas are thought to be caused by long-term exposure to ultraviolet (UV) radiation from sunlight[39].

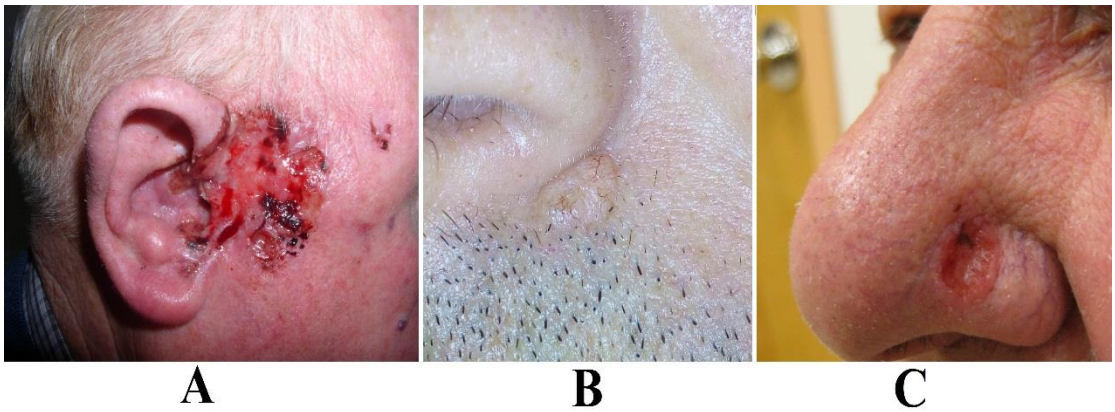


Figure 2.7: Basal cell carcinoma Initial and advanced status

Melanoma (mel): Malignant neoplasm derived from melanocytes that may appear in different variants. Melanomas are usually, but not always, chaotic, and some criteria depend on the site location. If excised in an early stage it can be cured by simple surgical excision. Melanomas can be invasive or non-invasive (in situ). And have different types[40] (Figure 2.8).

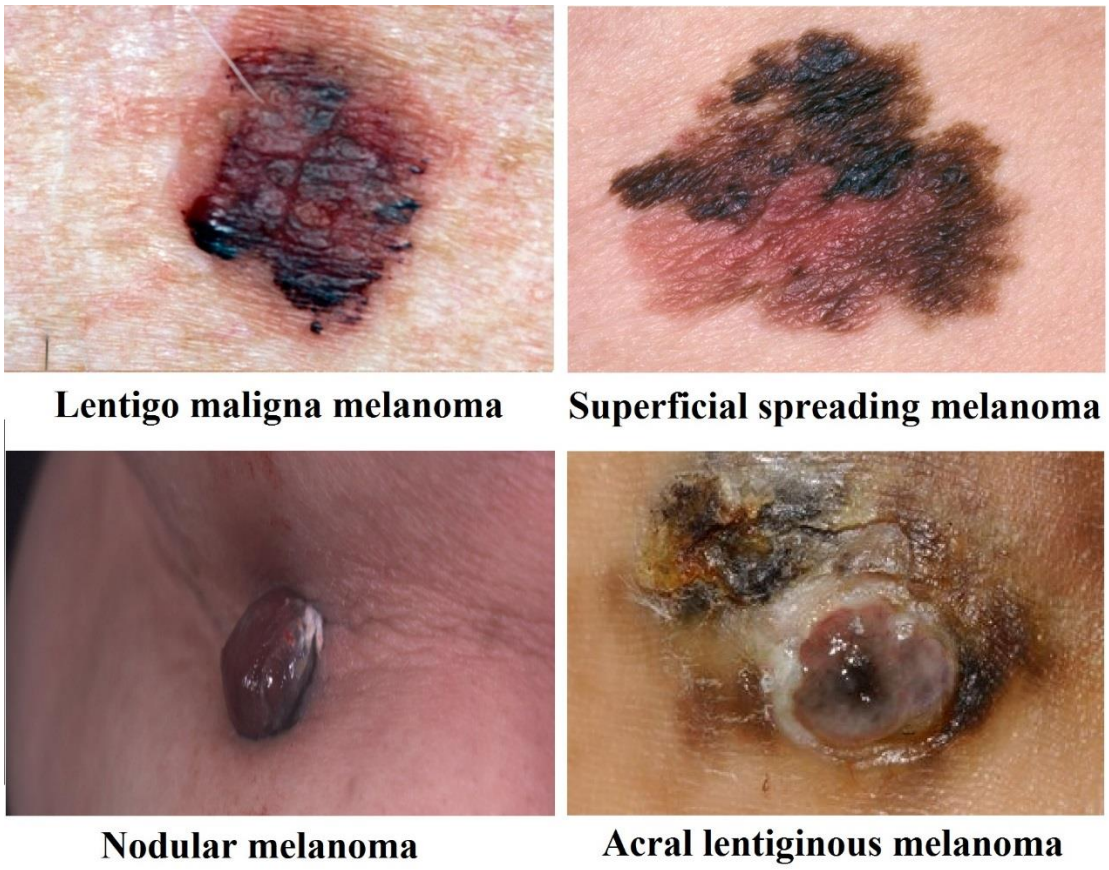


Figure 2.8: the four basic types of melanoma

Squamous cell carcinoma (SCC): is the second most common form of skin cancer. It's usually found on areas of the body damaged by UV rays from the sun (Figure 2.9). Unlike other types of skin cancer, it can spread to the tissues, bones, and nearby lymph nodes[41].

Figure 2.9: Various body parts infected with Squamous cell carcinoma

2.2.3 Appearance differences between skin lesions Benign vs. Malignant

There are specific patterns/features which may represent melanomas globally or locally. The global features allow a quick preliminary categorization of a given pigmented skin lesion prior to more detailed assessment, and they are presented as arrangements of textured patterns covering most of the lesion. The local features represent individual or grouped characteristics that appear in the lesion. Multicomponent pattern is a global feature that is most predictive for the diagnosis of melanoma, whereas the globular, cobblestone, homogeneous, and starburst patterns are most predictive for the diagnosis of benign melanocytic lesions. A typical pigmented network, irregular streaks, and regression structures are local features that show the highest association with melanoma, followed by irregular dots/globules, irregular blotches, and blue-whitish veil. On the contrary, typical pigmented network, regular dots/globules, regular streaks, and regular blotches are mostly associated with benign melanocytic lesions [42]. (Table 2.1) presents the general characteristics comparison between benign and malignant lesions.

Even with this known differences in the appearance of the benign and malignant lesions, but still hard to detect without professional clinical diagnostic due to appearance similarity (Figure 2.10) and high potential area where can the lesion develop (Figure 2.11).

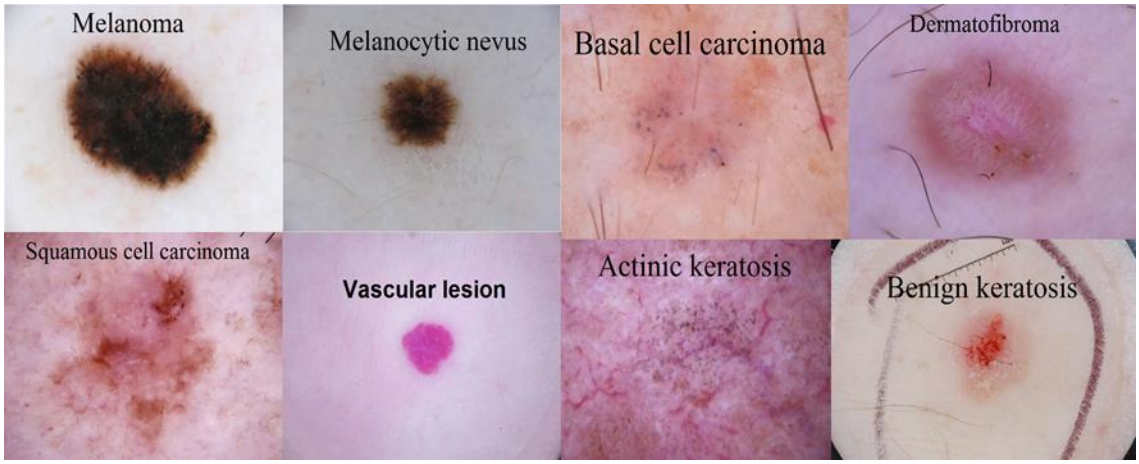


Figure 2.10: Samples of skin lesions from ISIC archive

Characteristic	Benign lesion	Malignant Lesion
Growth	Not growing	Growing-either slowly or rapidly
Bleeding	Absent	Present
Scabbing	No scab	Scab or keratin „crust“
Number/location	Many other similar lesions	On a sun exposed area of the body
Shape	Regular shape with smooth outline or line of symmetry	Irregular outline with no symmetry
Color	Uniform pigmentation	Variation in color throughout the lesion
Occurrence	Present for many years	New lesion

Table 2.1: Characteristics comparison between benign and malignant lesions [43]

2.3 Skin cancer (Melanoma example) stages

Stage 0 : Also called carcinoma in situ, cancer discovered in this stage is only present in the epidermis (upper layer of the skin) and has not spread deeper to the dermis.

Stage I : The cancer is less than 2 centimeters, about 4/5 of an inch across, has not spread to nearby lymph nodes or organs, and has one or fewer high-risk features.

Stage II : The cancer is larger than 2 centimeters across, and has not spread to nearby organs or lymph nodes, or a tumor of any size with 2 or more high-risk features.

Stage III : The cancer has spread into facial bones or 1 nearby lymph node, but not to other organs.

Stage IV : The cancer can be any size and has spread (metastasized) to 1 or more lymph nodes which are larger than 3 cm and may have spread to bones or other organs in the body.

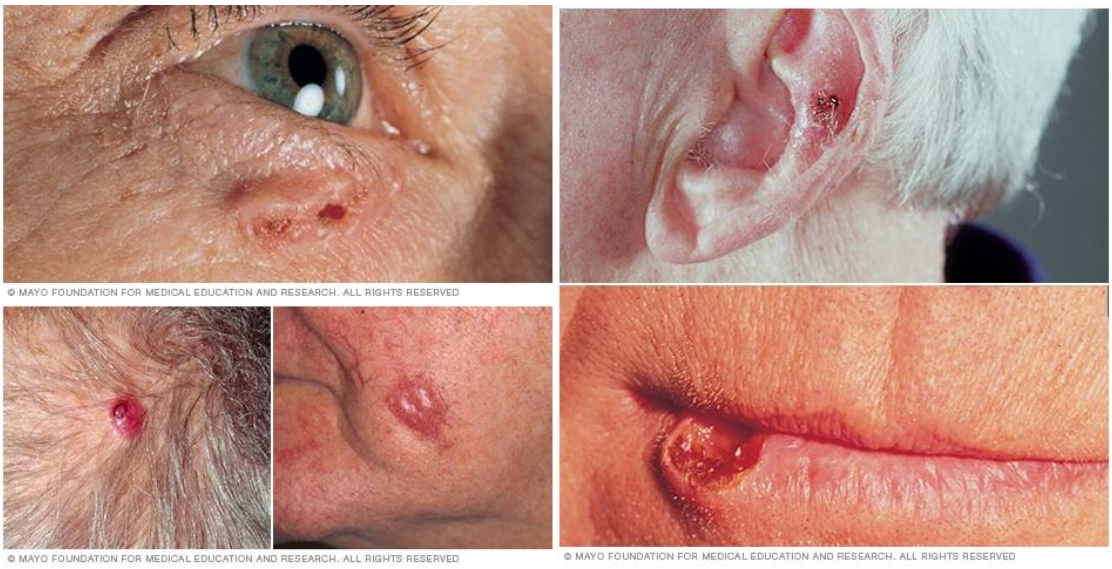


Figure 2.11: Samples where skin cancer can develop

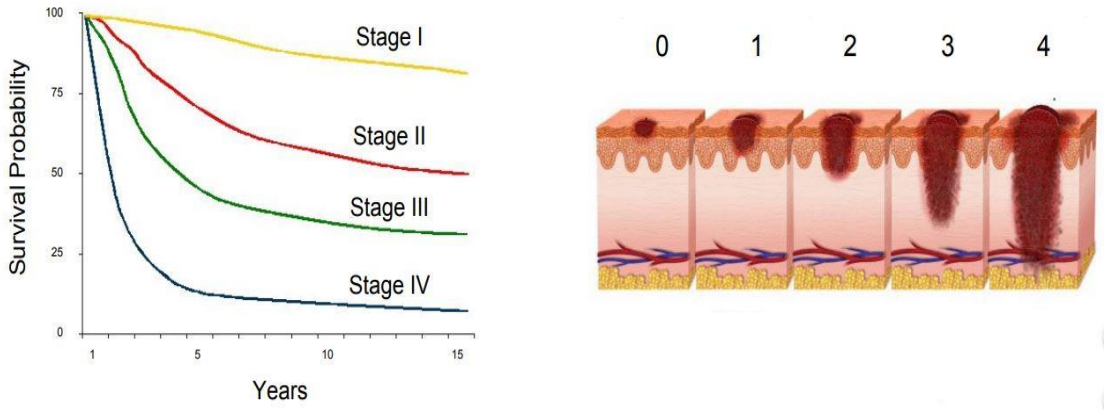


Figure 2.12: Skin cancer stages survival probabilities and development with time

Also of importance are the "Clark level" and "Bristow's depth", which refer to the microscopic depth of tumor invasion [44].

(Figure 2.12) represents melanoma stages [45] and the survival rates in 5 year:

Stage 0: Melanoma in situ (Clark Level I), 99.9% survival

Stage I / II: Invasive melanoma, 89–95% survival.

Stage II: High risk melanoma, 45–79% survival

Stage III: Regional metastasis, 24–70% survival

Stage IV: Distant metastasis, 7–19% survival

2.3.1 Clark's level

Clark's level is a related staging system, used in conjunction with Bristow's depth, which describes the level of anatomical invasion of the melanoma in the skin[46].

Five anatomical levels are recognized, and higher levels have worsening prognostic implications. These levels are:

- Level 1 : Melanoma confined to the epidermis (melanoma in situ)
- Level 2 : Invasion into the papillary dermis
- Level 3 : Invasion to the junction of the papillary and reticular dermis
- Level 4 : Invasion into the reticular dermis
- Level 5 : Invasion into the subcutaneous fat[47].

2.3.2 Bristow's depth

In medicine, Bristow's depth was used as a prognostic factor in melanoma of the skin. It is a description of how deeply tumor cells have invaded. Currently, the standard Bristow's depth has been replaced by the AJCC depth[48]. As shown in (table 2.2), Bristow's depth was originally divided into 5 stages[49].

Stage	Depth
Stage I	less or equal to 0.75mm
Stage II	0.75 mm - 1.5mm
Stage II	1.51 mm - 2.25mm
Stage III	2.25 mm - 3.0mm
Stage IV	greater than 3.0 mm

Table 2.2: Bristow's depth

The above studies showed that depth was a continuous variable correlating with prognosis. However, for staging purposes (table 2.3) shows the most recent AJCC guidelines[48] use cutoffs of 1 mm, 2 mm, and 4 mm to divide patients into stages.

Tumor Depth	Approximate 5 year survival
<1 mm	95-100%
1 - 2 mm	80-96%
2.1 - 4 mm	60-75%
>4 mm	50%

Table 2.3: Survival figures from British Association of Dermatologist Guidelines 2002

2.4 Existing clinical melanoma diagnostic methods

Clinically in the past, most physicians checked the skin moles by naked eyes and clinical experience. These days a magnifying instrument called dermoscope, that uses epiluminescence microscopy (ELM) images, is used to refine the diagnosis. Biopsy is recommended if something suspicious is found while diagnosis. Dermoscope could help dermatologists to assess and extract much more information from the skin lesion. Pattern analysis was the first dermoscopic method presented for diagnosis of pigmented skin lesions [28] and it has been further modified and refined by the International Dermoscopy Society (IDS). The other methods are ABCD rule, Menzies method and the 7-point checklist (7PCL). These methods attempt to simplify the pattern analysis method by analyzing only a small subset of dermoscopic structures and create a scoring system and this lowers their accuracy from the full system pattern analysis [49].

Figure 2.13: dermatoscope to the right and epiluminescence microscopy to the left

Pattern Analysis

Diagnosis based on pattern analysis needs critical assessment of the dermoscopic structures (features) that are seen in a pigmented skin lesion, since many patterns have dermoscopic structures. The first thing to do is check if there is a pigmented structure or not.

Benign and malignant melanoma has their own characteristic features that are discriminated by the colors, architecture, symmetry and homogeneity. (Table 2.4) presents the dermoscopic differentiations between benign and melanoma lesions using pattern analysis.

	Low	Medium	High
Colors: few versus many 1-light brown, 2-dark brown, 3-black, 4-red, 5-white, 6-blue (Score 1 point for each color)	1-2 colors (1-2 points)	3-4 colors (3-4 points)	5-6 colors (5-6 points)
Architecture: order versus disorder (Score 0-2 points)	None or mild (no points)	Moderate (1 point)	Marked (2 points)
Symmetry versus asymmetry border, colors and structures (Score 0-2 points)	Symmetry in 2 axes (no points)	Symmetry in 1 axis (1 point)	No symmetry (2 points)
Homogeneity versus heterogeneity pigment network, dots/globules, blotches, regression, streaks, blue-white veil, polymorphous vessels (Score 1 point for each structure)	1 structure (1 point)	2 of structures (2 points)	3 ≥ structures (3-7 points)

Table 2.4: Dermoscopic differentiations between benign and melanoma lesions using pattern analysis

Add up the score for a total of (2-17). The score of 7 or less is likely benign and the score of 8 or more is suspicious of melanoma [49].

ABCD rule

ABCD rule is a common feature extraction method, which helps dermatologists to recognize melanoma in its early stages. ABCD describes the clinical features of melanoma using parameters A (standing for Asymmetry), B (Border irregularity), C (Color) and D (Diameter) and these are stated as follows [49, 50]:

Asymmetry: If a lesion is symmetric (0 value) then it is benign (non-cancerous). For cancerous cases asymmetry in zero (value 1), or two orthogonal axes (value 2) are considered.

Border irregularity: The irregularity of a lesion indicates the existence of a cancer. To calculate the border, the lesion is divided into eight sections and the sharpest pattern will get the score of 1, its value ranges from 0 to 8 for the minimum and the maximum irregular borders respectively.

Colors: cancerous skin lesions pigmentation is not uniform. The presence of up to six known colors could be detected: white, red, light brown, dark brown, slate blue, and black. Its value ranges 0 to 6, a score of 1 for the presence of each of these colors.

Diameter: The diameter that is greater than 6mm are most likely to be melanoma than the small ones.

At the end, the Total Dermoscopy Score (TDS) based on the above four parameters, is computed as follows:

$$TDS = 1.3 * A + 0.1 * B + 0.5 * C + 0.5 * D$$

If the score is less than 4.75, then the lesion is benign, if it is between 4.75 and 5.45, then its suspicious to melanoma and if the TDS is greater than 5.45, then it is melanoma [50].

Menzies method

Menzies method classifies the dermoscopic features of benign melanocytic lesions from melanoma by two negative and positive feature sets. The negative set includes symmetry of pigmentation pattern and presence of only a single color and the positive set contains 9 positive features which are blue-white veil, multiple brown dots, pseudopods, radial streaming, scar like depigmentation, peripheral black dots/globules, multiple colors (5 or 6), multiple blue/gray dots and broad pigmentation. So for a lesion to be diagnosed as a melanoma it must have neither of both negative features and 1 or more of the 9 positive features [28].

7-point Checklist

The 7-point checklist assigns points to specific dermoscopic structures as per the checklist. It consists of 3 major features: atypical pigment network, gray-blue areas and atypical vascular pattern as well as 4 minor criteria: streaks, blotches, irregular dots and globules and regression patterns. When any of the major features is detected in a melanocytic lesion, immediate help from health professionals is recommended. When there are minor features, it is advised to be monitored regularly. The minor criteria are worth 1 point each whereas the major are worth two. Finally a total score is computed by summing the point value based on the presence of each criterion and if the score is greater than 3, then the lesion is classified as melanoma [49].

Chapter 3

3 Basics of Machine Learning, Neural Networks and Convolutional Neural Networks

Introduction
Artificial neural networks and especially convolutional neural networks are the main topic of this thesis. Therefore, it is important to get a good introduction into this topic. It is not possible to go deep into the matter. There are complete books which fill this topic, like the "Deep Learning" book from Goodfellow et al. [51]. So this chapter can only provide a shallow introduction to thesis topics. We start in this chapter with a short introduction to the basics of machine learning and continue with an introduction of artificial neural networks and convolutional neural networks.

3.1 Basics in Machine Learning

Mitchell [52] define machine learning with this definition:

> *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 the experience E.*

This is a little bit confusing definition, but after I introduce some examples for task T, performance measure P and experience E, it will be make more sense.

The task T describes the task which should be fulfilled with the computer program. This could be a classification, regression, clustering or some more complicated task, like driving a vehicle. In this thesis the task T is a classification problem, to classify a skin lesion into eight (8) classes.

The performance measure P is a measurement of the quality of the learned task. The performance measure P is depending on the task T. For classification it could be used the accuracy, which describes the ratio of correct classified examples with respect to all examples. In regression could it be the squared distance to the correct value, and in the task of driving a vehicle, it could be the average traveled distance until occurred the first failure. Sometimes it could be the measurement which penalize some mistakes not so hard as others.

With this measurement, we could evaluate the learned program or the model. Usually, we are interested in the performance of the model on unseen data. But therefore, the model must be evaluated on an independent test set. This test set is separated from the training data and is never used for the training. Then we could do an estimation of the performance on unseen data.

The experience E describes source of the data which is used to learn. In a classification and regression task, it is a data set with input data and the desired output label to the input label. In case of the driving a vehicle it is a recorded sequence of images and steering commands. In this thesis, the experience is a huge amount of images, which shows one lesion, and to every image exists one label, which describe the type of the tumor presented on the image.

A practical view of a machine learning system is depicted in (Figure 3.1). The process is split into two phases. In the first phase, the machine learning algorithm is used to learn from the training data, and the second phase is the prediction. The training data could be labeled images of skin lesions. The machine learning algorithm learns a model on these data and this model can then use it to predict unseen images of the same task. This prediction is happen in the second phase and apply only the learned model. In our example, the learned model get images of a skin lesion and must predict the label.

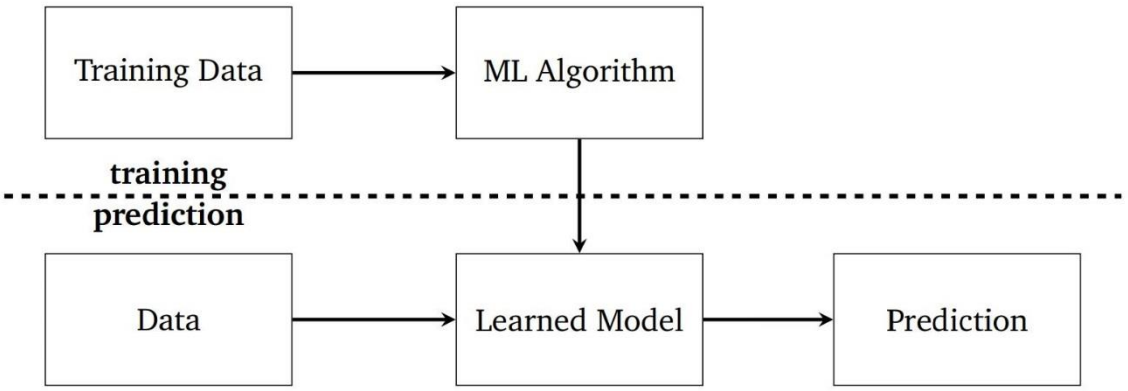


Figure 3.1: Workflow of a machine learning problem

3.2 Learning Problems

In machine learning it can be distinguished between different learning problems. The main learning problems are:

Supervised Learning: In supervised learning, the machine learning algorithm gets a labeled training set. The training set consists of several examples, and every example is a pair of input data and a labeled output data. The goal is to find a general function or rule that maps the input to the desired output label. Furthermore, the mapping should be general so that unseen data is also correct mapped[51].

Unsupervised Learning: In unsupervised Learning, the machine learning gets an unlabeled training set. The training set has only examples, but no label. The machine learning algorithm should find a structure in the data. This could be done with clustering methods. This means, that examples with similar properties should be grouped[51].

Semi-supervised learning: Semi-supervised learning is a class of machine learning tasks and techniques that also make use of unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning and supervised learning[51].

Reinforcement Learning: In reinforcement learning the machine learning algorithm interact with an environment and must reach a certain goal. This could be to learn to playing a game, or to driving a vehicle in a simulation. The algorithm gets only information how good or bad he has interact with the environment. For example in learning to play a game, could this information be the winning or losing of a game[51].

3.2.1 Under- and Overfitting

A machine learning algorithm must perform well on unseen data. This ability is called generalization. The generalization error is measured on a test set. If a model perform not well on unseen data, then there are two reasons. Either the model has not enough capacity and underfit the underlying function, or it has too much capacity and overfit the underlying function. If a model is underfitting, then both the training and test error will be high. If the model is overfitting, then the training error will be low and the test error will be high. The goal is to find a model, which has a low generalization error. The typical curves for training and generalization error are depicted in (Figure 3.2) [51].

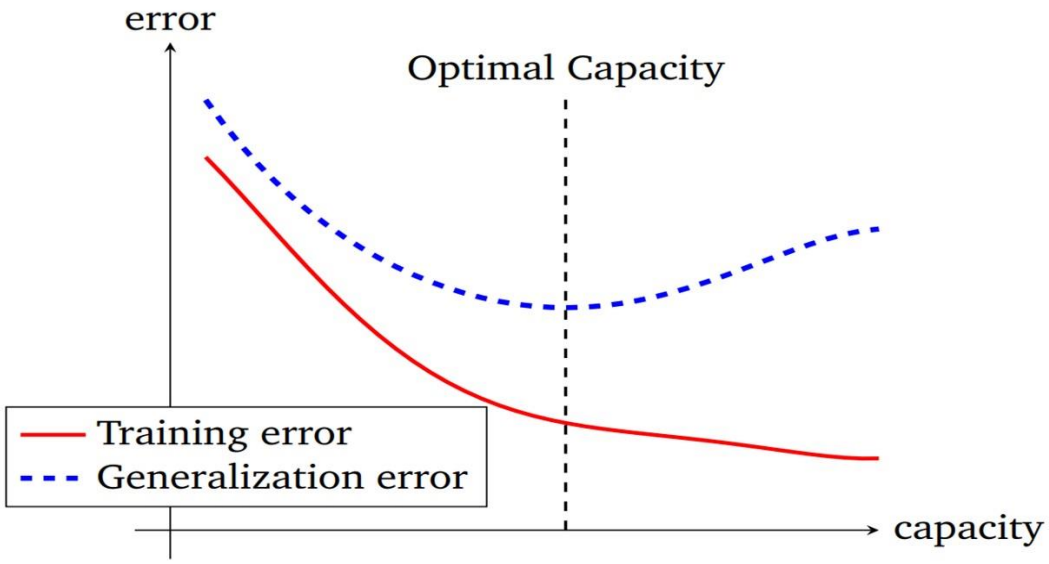


Figure 3.2: Shows typical curves for training and generalization error

In Figure 3.3 there are three diagrams depicted, which shows the same noisy sampling of a sinus function. The samples are used to train a model, which describes the underlying function. The left diagram is a model with a low capacity. So the training and the test error is high. In the center is a model with a higher capacity. This shows a good approximation of the underlying function. Nevertheless the model has a training error, because we sampled the sinus with noise. But this model will have on a test set the best generalization error compared to the other two models. The third diagram shows a model with a high capacity. The training error will be very low, but the learned model is not a good approximation of the sinus function, which would result on a test set with a high error.

It is also important to use the right capacity of a model for the problem. And if we add right regularization, then it is possible, that we can use a higher capacity [51]. Because the regularization damping the capacity of the model.

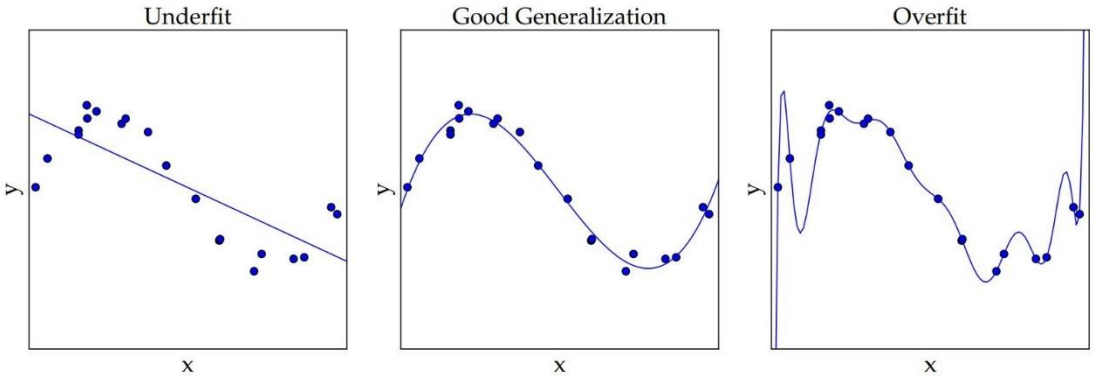


Figure 3.3: Performance diagrams of three models on the same sample points

3.2.2 Hyperparameters and Model Selection

Hyperparameters are parameters which belongs to the machine learning algorithm. With these parameters, the behavior of the algorithm can be changed, to be precise with some parameters the capacity of the model can be regulated. These parameters will be not learned during the training, but will be set by the developer at start of the learning process. In case of neural networks, this could be the learning rate, or the architecture of the model (number of layers or width of layers).

Normally, we test several models with different hyperparameters and chose the model with the lowest error. If we do this test only on the training set, then the lowest error is reached with the model, which has the highest capacity. This is not useful, as we saw in the overfitting section.

We want the model with the lowest generalization error. So the training set will be further split in a validation set. So that we have three data sets, a training, validation and a test set. The validation set is only used to measure the generalization error, and is not used for the training. The model with the smallest generalization error will then be used as the best model. But the predicted performance on the validation set has a bias, because we chose the model, which maximize the validation set. Therefore, the performance should be not measured on the validation set. That is why we have a third dataset, the test set. On this test set we could now predict the performance of the model, and this is a good approximation, how good the performance will be on unseen data.

3.3 Artificial Neural Networks

Artificial Neural Networks (ANN) – or short only Neural Networks (NN) – are inspired by neuroscience. It is the attempt of mimicking the biological neural network (BNN) of a brain, to solve problems in the same way. The research of ANNs are not new. They have a long history, it started in the 1940s. The breakthrough of ANNs occurs with the introduction of perceptron by c [74], which is the basis of ANNs to this day. For a good historical survey of NNs, I recommend the paper of Schmidhuber [54].

Over the time, the ANNs had different periods of popularity and was further development. The last period is today, under the name of Deep Learning. After the winning of different competitions with ANNs, the popularity of ANNs are increased in the research community. The main characteristic of deep learning are the larger networks – deeper and wider – which are used to solve a problem. Goodfellow et al [51]. see two reasons for the success of deep learning:

- More computationally power
- Larger datasets

Today, the computers have more computationally power to train a bigger network in shorter time. So it is possible to test different structures and different hyperparameters in shorter time. Furthermore, the larger datasets helps to find a better generalization.

I would like to add a third reason for the success of deep learning. The simplicity of applying deep learning on a problem. It is not more needed to have expert knowledge on the problem domain, to extract useful features. In the most cases, the raw data need only a little preprocessing step, before the data is used to train an ANN. The ANN learns the needed features from the raw data to fulfill the task. This simplicity and the good results on different areas, make the ANNs popular in our opinion.

The ANN contains a network of neural units. Typically the ANN is structured in layers. Every layer contains a different amount of neural units, which will be shortly called units. The units are an abstract model of biological neurons. Every unit is with other units connected.

This Section is only a short introduction to Neural Networks, and the information are taken from the chapters for Neural Networks from the books of Adamy [55], Goodfellow et al. [51], and Russell and Norvig [56], if not other cited.

3.3.1 Perceptron

The perceptron is an abstract model of a single neuron and was introduced by Rosenblatt [53].

the (Figure 3.4) is depicted a perceptron. The perceptron has multiple inputs $(x_1, x_2, \dots x_n)$, and only one output y. Furthermore, the perceptron has a bias input, which is called x_0.
For every input is derived a linear combination with the weights $(w_1, w_2, \dots w_n)$:

$$z = \sum_{i=0}^{n} x_i . w_i \tag{3.1}$$

During the training, the weights will be learned. So it is possible to reinforce or to weaken the input and get another output.

After the linear combination is computed, the value z is inserted into the activation function $\sigma(Z)$. This activation function activates the output, if the threshold is fulfilled. There are different activation functions with different properties, but for simplicity, the heavyside function is used:

$$\sigma(Z) = \left\{ \begin{matrix} 0: & if\ Z>0 \\ 1: & if\ Z \leq 0 \end{matrix} \right\} \tag{3.2}$$

With this activation function, the perceptron has the ability to linear split the hyperspace. This means, that the perceptron can only learn problems, which are linear separable. For many tasks, this is not sufficient. But with multilayer perceptron, it is possible to solve nonlinear problems, which is as next presented.

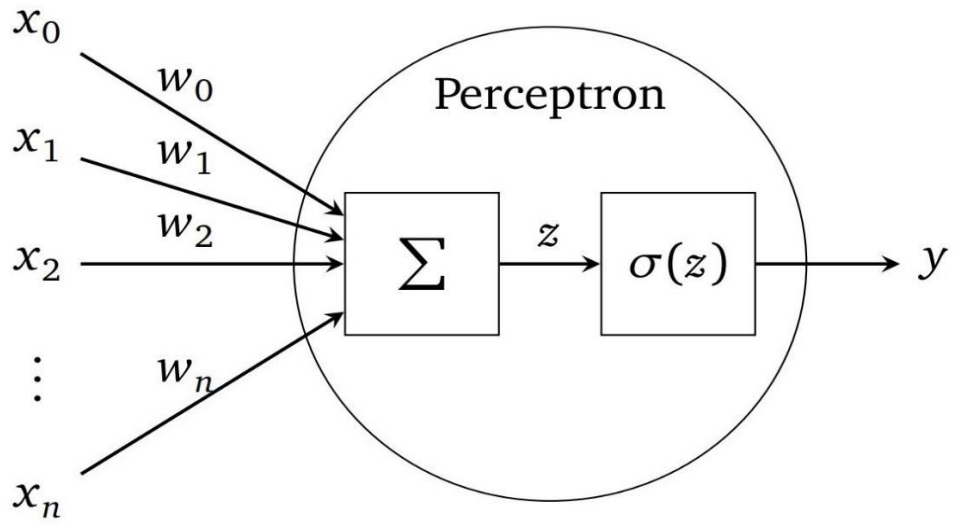


Figure 3.4: One perceptron with n inputs and one output

3.3.2 Feedforward Neural Networks

The feedforward neural networks or known under the name multilayer perceptron (MLP), are a multilayered networks of perceptrons. This means, that several perceptrons are connected in series. In (Figure 3.5) an example feedforward neural network is depicted. A feedforward neural network consists of multiple layers, which have specific names. The first layer is called the

input layer. The last layer is called the output layer. The other layers are the hidden layers. The hidden and output layer consists of several perceptrons, which are called units. The input layer contains only the values for the input. The depth of a network describes the number of layers, and the width of a layer describes the number of units. The depicted example network in (Figure 3.5) has the depth of three. The width of the hidden layers are four and the width of the input and output layers are two. The bias units are depicted with +1 as it's the common representation of the bias, and will be not count to the width of the layer[51].

Between every layer, the units are fully connected (FC). This means that every unit of layer '$L + 1$' has the output of all units from layer 'L' as input. The number of links and consequently the number of weights, will fast increase, if the width of the layers are increased. If layer 'L' has 'a' units and layer '$L - 1$' has 'b' units, then the number of weights $|W_l|$ between this layer are $|W_l| = \mathrm{a}.(\mathrm{b} + 1)$. The +1 comes from the bias input. So it is not uncommon that modern NNs have millions of parameter, which must be learned.

Through this concatenation of several layers, the feedforward neural networks gets the capability to solve non-linear problems, if the activation function is not linear, because the composition of linear function produce again a linear function. So the activation function should be a non-linear function. In Section 3.3.4 we show some useful activation functions.

Furthermore, a feedforward neural network has a very powerful property. With only one hidden layer with sufficient units and the right activation function, like the sigmoid function, the network can approximate arbitrarily closely every continuous functions on a closed and bounded subset of $\mathbb{R}^n$. This means that a network with a single layer has the ability to represent any function. But it is not guaranteed, that the training algorithm will find this function. It is possible, that the training algorithm is not able to find the right value for the parameters. Or the training algorithm choose the wrong function due to overfitting. Furthermore, the single layer may be infeasible large and it exists no heuristic how large the layer should be.

Nevertheless, deeper networks are empirical better and have a lower generalization error as shallow networks with one hidden layer. Goodfellow et al. [23] has showed empirical, that deeper networks are better on the recognition of house numbers in street view imagery. Nielson [57] see the reason for the better performance of deeper networks in the ability to learn a complex hierarchy of concepts. For example in the recognition of objects in images, the deeper networks learns in the lower layers edges and corners, and compose this information to more complex structures. In other words, in lower layers will be learned simple representations, which then are composed to more complex representations, up to an approximation of the searched function.

For the designing of NNs, it is practical to use a layer as a uniquely building block. So the layers of this section, will be called fully connected (FC).

Later we will present other types of layers, like convolution, pooling or dropout layer. Commonly, a layer has only one specific task. Furthermore, the layers could be seen as a function $f^i(x) = y$, which manipulates the input x to output y. So a feedforward neural network can be described as a composition of functions with:

$$f(x) = f^{(n)}\left(\ldots\left(f^{(2)}\left(f^{(1)}\right)\right)\right) \tag{3.3}$$

where $f\ (x)$ describes the complete network.

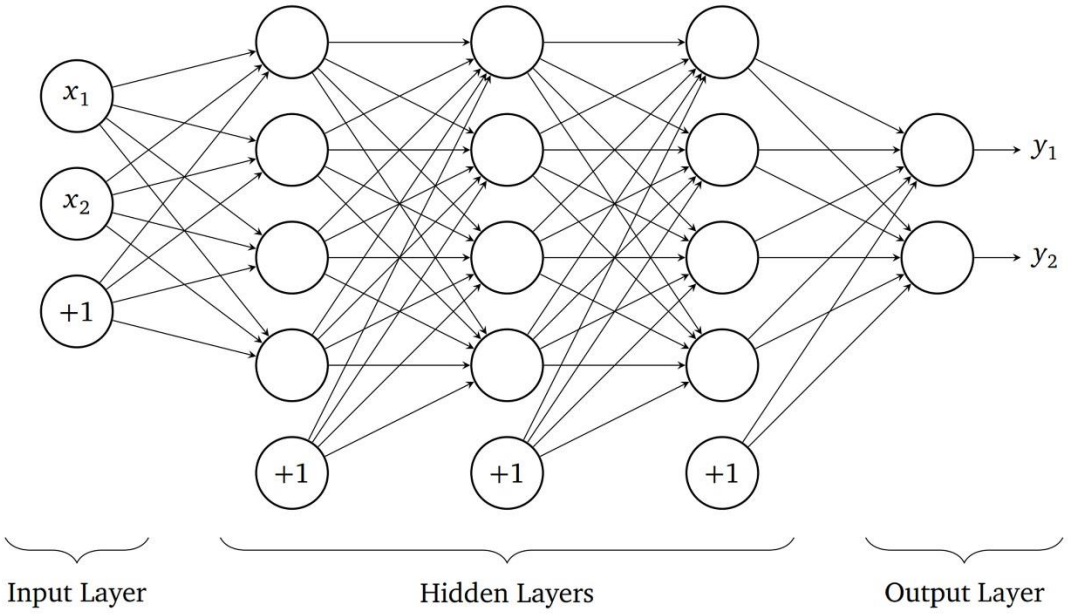


Figure 3.5: Multi-Layer Perceptron representation

3.4 The Training

So far, we have not described how the learning work for ANNs. In this section, we consider only the supervised training. This means, that we have a training set of input vectors $\{\tilde{X}_n\}$, where $n = 1, \ldots, N$, with a corresponding set of output vectors $\{\tilde{Y}_n\}$. The goal is training the function $f(\tilde{X}; \Theta) = y$ of the ANN so to adjust, that the function approximates the training data. Θ describes the parameter, which are learnable during the training. In our case, this are the weights on the links between the units. Therefore, it will be used a loss function. A simple loss function $J(\Theta)$ is the mean square error (MSE), which compute the squared difference between the $f(\tilde{X}; \Theta)$ and $\tilde{Y}$. Furthermore, the loss function computes the average loss over the complete training set:

$$J(\Theta) = \frac{1}{2N} \sum_{n=1}^{N} || f(\tilde{X}_n; \Theta) - \tilde{y}_n ||^2 \tag{3.4}$$

The loss function $J(\Theta)$ is used to minimize the error between the training data and the ANN with respect to Θ. For the minimization of the loss function, we will use a gradient descent method, which needs the gradient of the loss function with respect to Θ. To calculate the gradient, we use the backpropagation algorithm, which is later in this section introduced.

The gradient descent method is an iterative optimization algorithm, which updates the weights with the help of the gradient. In the simplest version, it has the following form:

$$\Theta \leftarrow \Theta - \eta \cdot \frac{\partial J(\Theta)}{\partial \Theta} \tag{3.5}$$

where η describes the learning rate. There are some modifications of the gradient descent, which will be later introduced. The gradient descent methods find not necessarily the global minimum. The gradient descent can find a local minimum or in the worst case it can stuck in a saddle point. Therefore, another initialization of the parameter Θ can results in a different solution.

The initialization of Θ and thus from the weights of the network, is important. Glorot and Bengio [58] showed, that a randomly initialization works not so good for deep networks, and they introduced a new heuristic. But before we show the heuristic, we define a notation for the weights in Θ. Every weight in Θ will be described by $w_{j,k}^{(l)}$, where 'l' is the layer, 'j' describes the unit in the layer and k describes the unit of the previously layer. For a better understanding, this is in Figure 3.6 depicted for the unit 'j' in layer 'l'. Let's go back to the heuristic of Glorot and Bengio. The heuristic uses the layer size n_{l-1} of the previously layer and the size n_l of the actual layer 'l'. To initialize the weights of layer 'l', they use the following initialization:

$$w_{j,k}^{(l)} = U\left(-\frac{6}{\sqrt{n_{l-1}+n_l}}, \frac{6}{\sqrt{n_{l-1}+n_l}}\right) \tag{3.6}$$

where $U[-a,a]$ describes a uniform distribution between $(-a,a)$, 'j' and 'k' describes the weights in the layer 'l'.

As next, the algorithm for the backpropagation will be introduced. After that the loss function for the classification problem, and then I will describe the modification of gradient descent for a faster convergence.

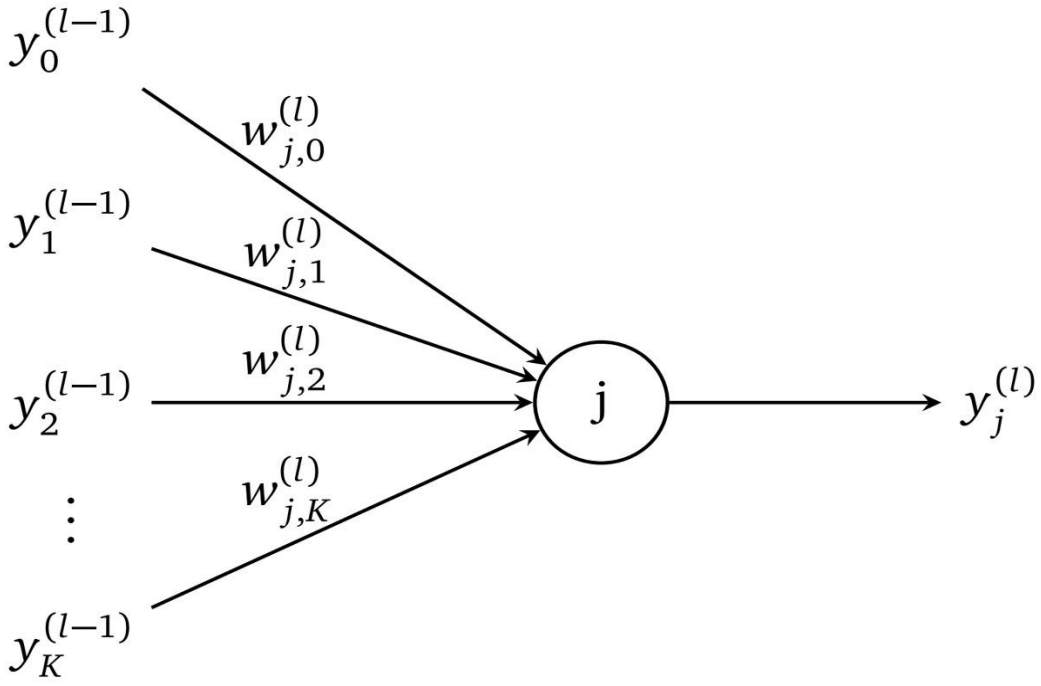


Figure 3.6: Shows one unit of the layer l. Image is adapted from [55]

3.4.1 Backpropagation

The backpropagation is an algorithm to propagate the error back from the loss function through the network to compute the gradient $\frac{\partial J(\Theta)}{\partial w_{j,k}^{(l)}}$. Backpropagation is not the learning algorithm of the ANN, but rather an efficient method to compute the gradient in respect to the weights. The learning algorithm is the gradient descent, which needs the gradient of the loss function, to update the weights. To derive for every weight the gradient, the backpropagation algorithm has the following recursively equation:

$$\frac{\partial J(\Theta)}{\partial w_{j,k}^{(l)}} = \delta_j^{(l)} \cdot y_k^{(l-1)} \tag{3.7}$$

With

$$\delta_j^{(l)} = \begin{cases} \frac{\partial J(\Theta)}{\partial y_j^{(l)}} \cdot \sigma_l'\left(Z_j^{(l)}\right), & \text{if } 'l' \text{ is the output layer} \\ \left(\sum_{i=1}^{q} \delta_j^{(l+1)} \cdot w_{i,j}^{l+1}\right) \cdot \sigma_l'\left(Z_j^{(l)}\right) & \text{, if } 'l' \text{ is a hidden layer} \end{cases} \tag{3.8}$$

Where $y_j^{(l)}$ describes the output of unit 'j' in layer 'l' and 'q' is the size of units in layer $'l+1'$. The function σ_l is the activation function of layer 'l' and $Z_j^{(l)} = \sum_{i=0}^{k} w_{i,j}^{l} \cdot y_i^{(l-1)}$ is the activation value from unit 'j' in layer 'l'. In (Figure 3.6) is for a better understanding of the values for one unit 'j' in layer 'l' the values depicted. Before the backpropagation process can be started, the forward propagation must be computed, to generate the activation values $Z_j^{(l)}$ and the output of $y_i^{(l)}$. These values are needed for the backpropagation. After the gradients is computed, the gradient descent step can be applied, with

$$w_{i,j}^{l} \leftarrow w_{i,j}^{l} - \eta \cdot \frac{\partial J(\Theta)}{\partial w_{j,k}^{l}} \tag{3.9}$$

And then the next iteration can be started. This is repeated, until an abort criterion is reached. This could be a maximal number of iterations, or that the loss is smaller than a predefined value.

3.4.2 Loss Function

In the Equation 3.4, we introduced the MSE loss function. This is one of many loss functions, which can be used. A loss function maps values of one or more variables to a single value of $\mathbb{R}$, and this value represents the loss. The loss describes the discrepancy between the function $f(\tilde{X}; \Theta)$ and the target value $\tilde{Y}$. The learning of the neural network is the reducing of the loss, which are described by the loss function. Therefore, the loss function $J(\Theta)$ is minimized with respect to Θ, which are the parameters of the neural network. It exists different loss functions and the selection of the loss function depends on the learning problem. In regression problems, the MSE loss function is a good choice. But for the problem of classification it is not so practicable. Therefore, there exists a common loss function for multi-class problems, called cross-entropy loss or also known as log loss. It calculates a loss between two distributions, with

$$L(p, q) = -\sum_{i=0}^{n} p_i \cdot ln\, q_i \tag{3.10}$$

where 'p' is the target distribution and 'q' is distribution which is computed by the neural network. To understand why the output of the network is now a distribution, we do an excursion to the output encoding for classification. For classification, the output is so encoded, that every class has its own output, which is represented in a vector of 'q'. The target label is also encoded

as a vector 'p' with the property, that only one entry is '1' and all other are '0'. This is called a one-hot vector. The classes are enumerates, so that every class represents an entry in the vector. This means, that the class 'i' is represented in the vector 'p' at index 'i'. The vectors 'p' and 'q' are then a distribution, if the sum of all the entries are '1', and all entries are ≥ 0. For the one-hot vector 'p', it is easy to see, that this property is fulfilled. But the computed vector of the neural network has not necessary this property. Therefore, it is used the softmax activation function at the output layer to get a distribution for the output. It changes the output so, that the required properties are fulfilled. We introduce the softmax activation function later in Section 3.4.5. The learning goal is now to minimize the loss between this two distributions. The nice side effect is, that the distribution 'q' gives us the percentage how sure the neural network is, that the prediction is this classes.

After we introduced the output encoding, we would like to add a note to the Equation 2.10. This is not the final loss function $J(\Theta)$, because this describes only the loss of one example in the training set. How in Equation 2.4 for the MSE, the average is computed of the cross entropy loss over all training examples.

3.4.3 Gradient Descent

We mentioned, that the Equation 2.5 for the gradient descent is a simple variant of gradient descent. For a better overview, we show the equation again:

$$\Theta \leftarrow \Theta - \eta \cdot \frac{\partial J(\Theta)}{\partial \Theta} \tag{3.5}$$

The gradient descent algorithm is an iterative optimization algorithm to find a local minimum of a function. It can be illustrated with the following scenario. We are on a mountain and want back to the valley. On the mountain is the visibility extremely low, because it is very foggy. Therefore, we see not the path to the valley and have only local information to find a path down to the valley. So we search the steepest descent at the current position. We go a step of size 'η' in the direction of the steepest descent and repeat the procedure, until we reached no improvements.

In analogy, the valley is the searched minima, but the path to the valley is unknown. The search of the steepest descent is the computation of the gradient. And the update of the parameter, is the step which we are going down with the step size 'η'. It is not ensured, that we reached the global minimum. The valley could be only a local minimum.

The choice of the learning rate 'η' is important, because if we use a too small step size, the algorithm needs very long time and it could stuck in a local minima. If 'η' is too big, it could happen, that we over jump the valley with the best solution. So it is one of the main hyperparameters of a neural network, and should be selected very carefully The starting point is also important, because another starting point results in a different path, which could find a better (local) minima.

3.4.4 Stochastic Gradient Descent

For the gradient descent exists some interesting extensions. The most important extension is the stochastic gradient descent (SGD) with mini-batches. It does the gradient descent iteration with a small subset of examples from the training set, instead the complete training set. The normal gradient descent is very unpractical regarding the computation time and update speed. For one update, it processes the complete training set, which can have millions of examples. This means, it must compute millions of forward propagations, before it can compute one

backpropagation step for the update of the weights. With SGD with mini-batches, it does with a small size of examples a weight update, but with this huge amount of updates it will be find the right direction, which minimize the loss. Some of the updates will be go in the wrong direction, but if the majority goes in the right direction, then the loss will be minimized. The size of the mini-batches ranging from one to a few hundred. With a low mini-batch size, the training time is increased enormously, because for every example is computed a forward propagation and a backpropagation, and the learning rate 'η' must be also small, because the gradient has a high variance. So the mini-batch is mostly set to a higher value with a power of two. But it is very interesting, that the best results are often reached with a mini-batch size of one [51]. And that a huge mini-batch size (over 1.000) result's in a (bad) sharp minima [59]. It is common, that the mini-batch size is only called batch size. At this point, it is appropriated to introduce two common terms in the training of neural networks:

Iteration: One iteration describes one update with the gradient descent. In other words, it is one learning step.

Epoch: One epoch describes one pass through the complete training set. Normally, one epoch consists of several iterations, because the size of the training set is much bigger than the batch size.

3.4.5 Activation Functions

The activation function $\sigma(z)$ is the main part of a neural network, which gives the network the power to solve non-linear problems. If the units have no activation function or only a linear activation function, then the neural network could only solve linear problems. No matter how deep the neural network is. So the activation function should have a non-linear characteristic. A further property for the activation function is, that it must be continuously differentiable. Otherwise, the backpropagation algorithm will not work, because it cannot compute the gradient.

In the beginnings of the research of neural networks, there were used saturated activation functions. This means that the activation was limited to (0,1) or (-1,1). As activation functions were used the sigmoid or tanh function. Both have the problem, that by small or big values of z the value of the gradient goes to zero, and the convergence of the learning decreases. In other words, the training of the neural network needs much more time. Another problem is the vanishing or exploding of the gradient, if the network is very deep. The vanishing gradient occurs in earlier layers, because through the backpropagation will often be multiplied with small values. Many multiplications with small values between '0' and '1' tends toward zero. The exploding gradient, can happen, if the weights are big, and the activation is near '0'. At this point the derivative of sigmoid and tanh has reached their maximum. With the time, new activation functions are developed, like the ReLU or ELU. In this section, we want to show two activation functions, which are used in this thesis. These are ReLU, and Softmax.

3.4.5.1 ReLU - Rectified Linear Unit

Actually, the Rectified Linear Unit (ReLU) is the most used activation function for neural networks. The formula for this function is simple

$$\sigma(z) = \max(0, z) = \begin{cases} 0, \text{if } z < 0, \\ z, \text{if } z \geq 0 \end{cases} \tag{3.11}$$

This means, that the function is 0, if z is negative otherwise the value is z. The ReLU activation is in Figure 3.7 depicted. The interesting fact is, that the function is for positive values linear and not saturated. A further property, which makes ReLU so popular is the fast computation of the derivative. The derivative is either 0 (at negative values) or 1 (at positive values). Through this, the vanishing of the gradient is not more a problem, because a multiplication with 1 doesn't change the error. Glorot et al. [60] showed, that ReLU has without pre-training better results as tanh or softplus. Furthermore, Krizhevsky et al. [21] have applied

ReLU in combination with Convolutional Networks and have a 6 time faster convergence as with the same network with the activation function tanh. Batch normalization with ReLU is heavily used in modern network architectures [61, 62, 63, 64]. We introduce batch normalization in the next section, but ReLU with batch normalization increases the learning speed of the network. This comes from the normalization of the batch normalization, so that the mean is zero. It is known, that the input with zero mean increases the convergence [65].

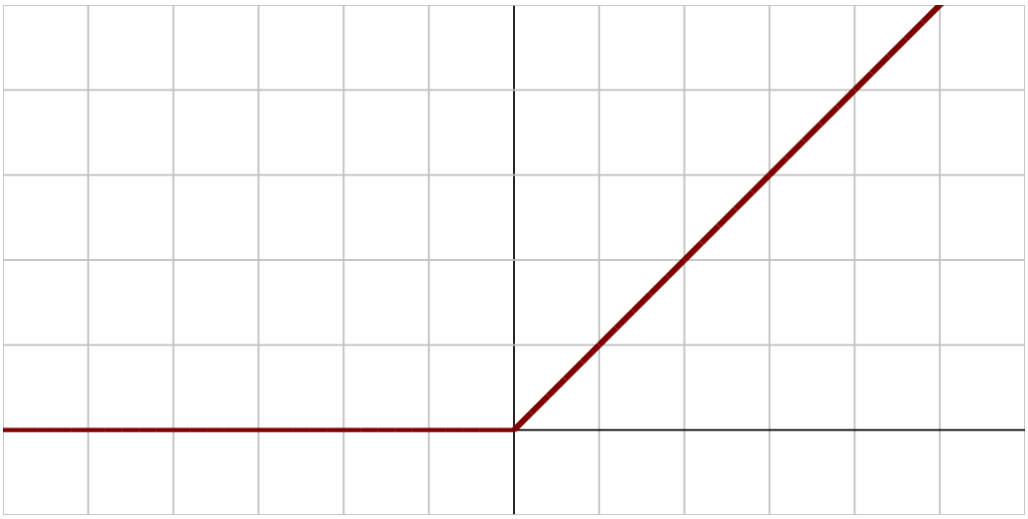


Figure 3.7: Diagram for the activation functions of ReLU

3.4.5.2 Softmax

In the section about the loss function, we have used the softmax activation for the encoding of the output. This is the main application of the softmax activation function, to represent a probability distribution over 'K' classes, and therefore it is only used in the output layer. The softmax is applied on the complete output layer and need all linear combination $\mathrm{z} = [\mathrm{z1}, \mathrm{z2}, \ldots, \mathrm{zK}]^T$ of all units. The formula for softmax is the following

$$\sigma(\mathrm{z})_j = \frac{e^{zj}}{\sum_{k=1}^{k} e^{zk}} \text{ , for j } = 1, \ldots, \mathrm{K}. \tag{3.12}$$

It scales every output of $\sigma(\mathrm{z})_j$ to a range between 0 and 1, and the sum of $\sum_{j=1}^{k} \sigma(\mathrm{z})_j = 1$. This represents a probability distribution. With the exponential function, the largest value in 'z' will be highlighted and the other values will be suppressed. For example, let $z = [1, 2, 3, 2, 1]^T$, then is the activation $\sigma(\mathrm{z}) = [0.07, 0.18, 0.50, 0.18, 0.07]^T$. The value 3 get much more of the available space, as the other values. If we would only scale the vector z to one, with a division of the length of the vector 'z', we would get $\frac{z}{|z|} = [0.11, 0.22, 0.33, 0.22, 0.11]^T$. Therefore the softmax gives the largest value in 'z' a bigger reward.

3.5 Regularization

Regularization are methods to control the overfitting or rather to improve the generalization error. There are different methods, how to apply regularization. Some of the methods are common approaches in machine learning, and other are only usable for neural networks. Goodfellow et al. [51] argument, that large models, that has been appropriately regularized, find the best fitting model. At this point, we show four regularization methods, which are present in this thesis.

3.5.1 L2-Regularization

L2-Regularization is a regularization methods, which is added to the loss function $J(\Theta)$. It penalizes the weights 'w' from Θ, but not the bias weights. $\Omega(\Theta) = \lambda\frac{1}{2}||W||^2$ is added to the loss function as its the regularization term . This results is a new loss function

$$\tilde{\boldsymbol{j}}(\Theta) = \mathrm{J}(\Theta) + \Omega(\Theta) \tag{3.13}$$

which is used to minimize the loss of the neural network. Furthermore, it is added a further hyperparameter 'λ', which represents the strength of the regularization. But what does this regularization? Through the adding of the quadratic weight to the loss function, the weights with a high value penalized, because they increases the loss. So, weights with a high value are bad for the loss. In other words, the L2-regularization brings the weights closer to the zero.

3.5.2 Early Stopping

On training of large models, normally the training and validation error decreases over the time, but at one point the validation error starts to increase. At this point the model is starting to overfit, and learn specific properties of the training set. To stop at this point, it is applied the method of Early stopping. It returns the model, which has the lowest validation error. Therefore, the training needs a validation set, to evaluate periodically the validation error. The normal period is after every epoch. But how could we ensure, that this increasing was not caused by noise? The training is not stopped after the first increasing of the validation error. The network is further trained until a threshold of "number of epochs without improvements" is reached. Through the evaluation of further epochs, we get the trend of the validation error for more training. For example, if 10 times in a row, the validation error has no improvements compared to the best validation error, then the training stopped, and the model with the best validation error is returned.

3.5.3 Batch Normalization

The training of a network changes the weights on every layer. This change has the effect, that the input distribution changes during updates of previously layers. The authors of batch normalization [65] called this effect internal covariate shift. To counteract the change of the distribution during the learning, they introduced the batch normalization. Every mini-batch, which is inserted in the network, will be on every layer normalized on the input of the previously layer. The formula for the normalization is the following

$$\mu \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.14}$$

$$\sigma^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \tag{3.15}$$

$$\tilde{x} \leftarrow \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{3.16}$$

where μ and σ describe the mean and the standard deviation, $\tilde{x}_i$ is the normalized value of the input. The value m describe the size of the mini-batch.

The value $\tilde{x}_i$ has then a mean of '0' and a standard deviation of '1'. The batch normalization will be applied after the linear combination of a unit. To not lost the representation power of the units, the value $\tilde{x}_i$ will be scaled and shifted

$$y_i \leftarrow \delta\tilde{x}_i + \beta \tag{3.17}$$

where δ and β are learned parameter during the training. So the value y_i could have any mean and standard deviation. This is useful in the case of using sigmoid as the activation function. Without the scaling and shifting, the activation function will be act as an almost linear

function, because through the normalization, the input is scaled to a range, on which the sigmoid function is almost linear.

At test time, μ and σ are replaced with the average, which is collected during the training. So it is possible to predict a single example, without to have a minibatch.

Through the applying of batch normalization, the learning rate can be increased, and this results in a faster training. Furthermore, the accuracy is increasing compared to the same network without batch normalization [65].

The batch normalization helps the activation function ReLU to learn faster and have a better performance, but on the activation function ELU there are no difference between with or without batch normalization [66].

3.5.4 Dropout

Dropout was developed by Srivastava et al. [67] and is a regularization method for neural networks. The key idea is, that units will be randomly dropped for every iteration of the training. This means, that the dropped units and their connection are zero. This should prevent the co-adapting of the units. During the testing the dropout is not available, and all units are used for the predicting. Srivastava et al. [67] tested dropout on different datasets and they had on all an improvement of performance. The drawback of dropout is the increasing training time, because not all weights a trained in one iteration. If one unit is dropped, then are all depending gradients are zero and therefore the corresponding weight will be not updated. Dropout could be seen as a training of a subset of the model. Every iteration builds a different version of the model, and every weight is updated with another set of weights. This prevents the co-adapting of the units, because a unit cannot more rely that another unit is available. A similar dropout is the Spatial Dropout, which is introduced by Tompson et al. [68]. It is used in convolution neural networks to drop complete feature maps and not only single units. This brings us to the next section, where we introduce convolution neural networks.

3.6 Convolutional Neural Networks

Convolution Neural Networks (ConvNets) are specialized Artificial Neural Networks. ConvNets are well suited for the processing of images, but the same concepts are adaptable for other fields, like audio or video. In this section, we describe the ConvNets for the processing of images. A ConvNet consists of multiple convolution and pooling layers. At the end follows normally a fully connected layer. A pooling layer is applied after one or multiple convolution layers. The convolution layers have the task to extract useful features from the input, which results in multiple feature maps. The pooling layer reduces the spatial size of these feature maps. In image processing is often applied convolution on images. It manipulates an image or extracts features from an image. Hence, there are different filters with different kernel sizes. For example, the Sobel filter, which has a kernel size of 3×3 size, will be applied with convolution on an image. The result is an image, which shows the edges of the original image. This idea of filters for feature extraction is translated to ConvNets. But instead of using hard coded filters, the ConvNet learns these filters. But to apply this on neural networks, it must be changed some structures on the neural networks. In the next sections, we will describe the convolution and the pooling layer. The information for the ConvNets are taken from the book of Goodfellow et al. [51] and the Stanford Lecture CS231n [69].

3.6.1 Convolution Layer

A convolution layer consists of units as well as a normal neural network, but with a different arrangement and a different connectionism of the units. The main differences to the neural networks are:

- Three dimensional arrangement of the units, instead of 1 dimension
- Weight sharing
- Local connectivity

The three dimensional arrangement comes from the image. A colored image has normally three channels RGB (red, green, blue), and every channel is described by a two dimensional matrix. Therefore, the input of the ConvNet is a three dimensional matrix. The output of a convolution layer is again a three dimensional matrix, with feature maps of two dimensions and this x times for the number of filters for this layer. Every filter produces a feature map. Weight sharing means, that the same weights are used for multiple output units. Through this, the ConvNet gets the property, that the features are invariant against translation. This means, that a feature can be found on the complete input. To compare this with the Sobel filter, the weights are the filter, and this filter will be used on the complete input to generate the output.

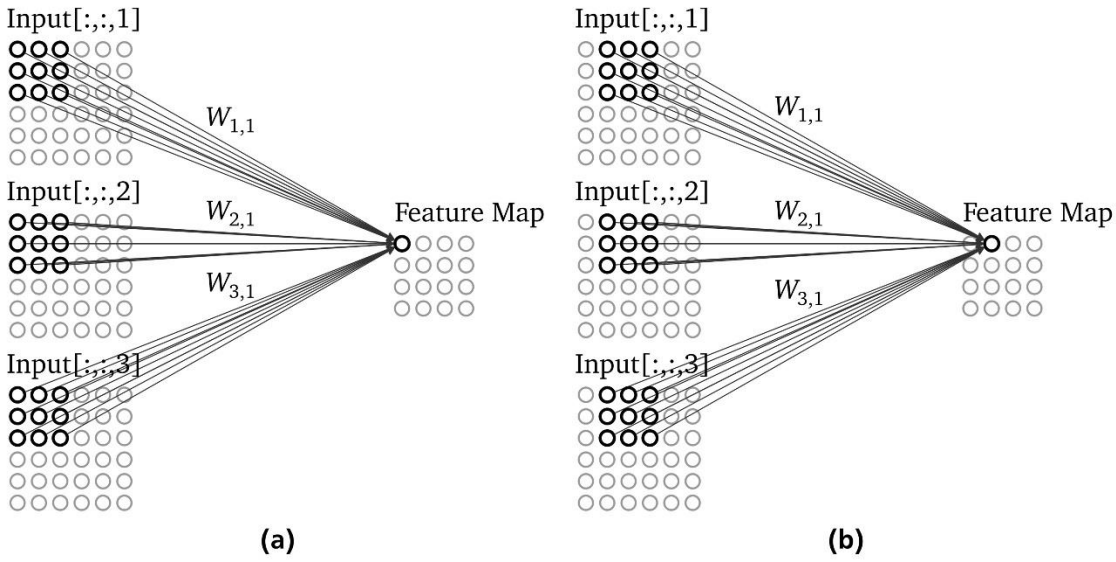


Figure 3.8: The connection between the input and output units of the convolution layer

Furthermore, the convolution layer has only one feature map for the output. The activation function and the bias are omitted for a better overview. The local connectivity is depicted in the two figures, with a kernel size of 3×3. (a) and (b) shares the same weights for the computation of the feature map. For another feature map, it would be used new weights. $W_{\mathrm{i,o}}$, describes a 3×3 weight matrix, which is used between the input channel 'i' and the output channel 'o'. The output channel is called feature map. The idea for this representation is from [57].

Local connectivity means, that not all units of the input are connected with the output unit. The size of the local connectivity is described by the kernel size.

To get all this together, in (Figure 3.8) is provided an example. The two figures show the connection between the input and two output units of the feature map. The input is showed with three channels. This could be the color channels R,G,B of an image, or the feature maps of a previously layer. The input is described by a three dimensional matrix. As output is depicted one feature map. For more feature maps is the behavior similar. The activation function and the bias are omitted for this example, but would be also applied on the units. The bias is also shared, which means, that one bias weight is used for all units in one feature map. The weights $W_{\mathrm{i,o}}$ are a 3×3 matrix, which describe one kernel, between $Input[:,:,i]$ and the $FeatureMap[:,:,o]$. The weights $W_{:,\mathrm{o}}$, describe one filter and are shared for one feature map. This means, that the same weights are used for the computation of all units in one feature map.

The spatial size of the feature maps are smaller than the spatial size of the input channels. In our example is the input spatial size 6×6, and the feature map only 4×4. This comes from the kernel size, because the kernel could be only applied on 16 different positions in the input channel. To avoid this effect and get the same spatial size for the feature map, the input must

be extended with a zero-padding. The number of units for the output, depends on the spatial size of the input and the number of filters. The number of filters describe how much feature maps should be generated. But the spatial size of the feature map and so the number of units, depends on the input spatial size.

Through the weight sharing and local connectivity, the number of weights decreases exponentially compared to a fully connected layer with similar number of units. Let K the width and height of a kernel, 'I' the number of input channels and 'F' the number of filters, which is equivalent to the number of feature maps. Then has the layer only $k^2 \cdot \mathrm{I} \cdot \mathrm{F}$ number of weights. In our example of (Figure 3.9), is $K = 3$, $I = 3$ and $F = 1$. Hence, the number of weights are $3^2 \cdot 3 \cdot 1 = 27$. In a fully connected network, the number of weights will be $6 \cdot 6 \cdot 3 \cdot 4 \cdot 4 = 1728$, which is much more!

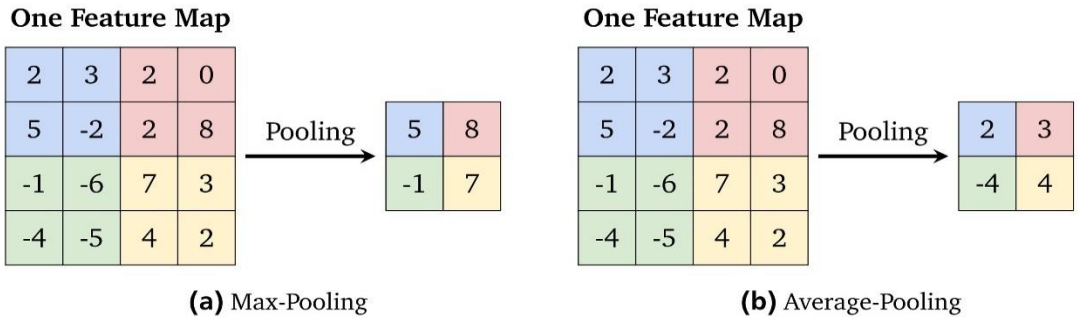


Figure 3.9: Example for the max-pooling and the average-pooling with a filter size of 2×2 and a stride of 2×2. Adapted from the Stanford Lecture CS231n [69]

Another parameter for a convolution layer, besides the number of filters and kernel size, is the stride. The stride is the size of sliding over the input channel. In our example, we have a stride of (1,1), and the local connectivity is slide one unit to right or down, for the next output unit. If we increase the stride to (2,2), then we would slide the local connectivity two units to the right or down, to calculate the next output unit. This has the effect, that the spatial dimension is reduced for the feature map.

But where is the convolution happened? From the previously point of view, it is not obvious where the convolution is applied. But mathematically, the convolution is applied between $Input[:,:,i]$ and the weights $W_{\mathrm{i,o}}$ Let us formulate, that $\mathrm{Input}[:,:,\mathrm{i}] = V_i$ and V_i is a two dimensional matrix, which describes the i-th channel from the input. And Z_0 is also a two dimensional matrix, which describes the o-th output channel or called feature map. $W_{\mathrm{i,o}}$ describes a kernel with the size of $k \times k$. Then is the o-th feature map calculated with

$$Z_0 = \sigma\left(\sum_{i=0} V_i * W_{\mathrm{i,o}} + b_0 1\right) \tag{3.18}$$

where the asterisk $'*'$ denotes the convolution, σ the activation function and b_0 the shared bias for the feature map, which is described by a scalar value. The above formula is not a scalar! Behind $V_i, W_{\mathrm{i,o}}$ and 1 are 2D-matrices. The discrete 2D-convolution is defined for two matrices I, K at point (i,j) with

$$(\mathrm{K} * \mathrm{I})_{i,j} = \sum_m \sum_n I_{\mathrm{i-m,j-n}} K_{m,n} \tag{3.19}$$

The convolution in a convolution layer are multiple convolutions, which are added up, which we can see in Equation 2.22. Without the addition of the convolutions of the input channels, we would not get a combination of features, which are essential for a ConvNet.

3.6.2 Pooling Layer

The pooling layer is normally applied after some convolution layers. It has the function to reduce the spatial size of the feature maps. This reduce the amount of parameters and the computation time, because the next layer get a smaller version of the feature maps. In other words, it down samples the feature maps. With one pooling with the stride size of 2×2, the spatial dimension of the feature maps is reduced of 75%. The pooling works independently on every feature map and has no activation function or weights to learn. Furthermore, the pooling layer helps to be invariant to small translations of the input, because a small translation has mostly no change on the values of the outputs of the pooling layer.

The most common pooling layers are the max- and average-pooling. Both pooling layers take as hyperparameters the filter size and the stride. The max-pooling computes the maximum, which lies in the receptive field of the filter, and the average-pooling computes the average. An example for both pooling are depicted in (Figure 2.10). As recommend values for the filter size and the stride are usually for both the size of 2×2. A stride of 1×1 has no down sampling character, and therefore it is not common. A higher size for the stride has normally a negative effect, because the reduction is too much (the filter size has at the minimum the same size like the stride). A filter size of 3×3 and a stride of 2×2 has sometimes a performance gain [21].

If the feature map is not divisible by the stride without remainder, then it will be cut off the last row or column, and we lost information. In this case, it is better to add a padding. Another approach to avoid this problem is to use an input size of power of two for the network. And add to all convolutions a padding. So will be all feature maps divisible by the stride size, if the stride was 2×2.

There are further approaches for the pooling, like to use a pooling with $max + average$. Another approach is to use no pooling, and instead to use a convolution with a stride [70]. This reduce also the size of the feature maps, and has the benefit, that a further non-linearity is used.

3.6.3 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.

Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task.

Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained.

Transfer learning only works in deep learning if the model features learned from the first task are general.

3.6.3.1 How to Use Transfer Learning

in computer vision, Neural Networks usually try to detect edges in their earlier layers, shapes in their middle layer and some task-specific features in the later layers. With transfer learning, you use the early and middle layers and only re-train the latter layers. It helps us to leverage the labeled data of the task it was initially trained on.

In Transfer Learning, we try to transfer as much knowledge as possible from the previous task, the model was trained on, to the new task at hand. This knowledge can be in various forms depending on the problem and the data. For example, it could be how models are composed which would allow us to more easily identify novel objects.

3.6.3.2 The benefits of using Transfer Learning

Usually, you need a lot of data to train a Neural Network from scratch but you don't always have access to enough data. That is where Transfer Learning comes into play because with it

you can build a solid machine Learning model with comparatively little training data because the model is already pre-trained. This is especially valuable in Natural Language Processing (NLP) because there is mostly expert knowledge required to created large labeled datasets. Therefore you also save a lot of training time, because it can sometimes take days or even weeks to train a deep Neural Network from scratch on a complex task.

Lisa Torrey and Jude Shavlik in their chapter on transfer learning [81] describe three possible benefits to look for when using transfer learning:

Higher start. The initial skill (before refining the model) on the source model is higher than it otherwise would be.

Higher slope. The rate of improvement of skill during training of the source model is steeper than it otherwise would be.

Higher asymptote. The converged skill of the trained model is better than it otherwise would be [81].

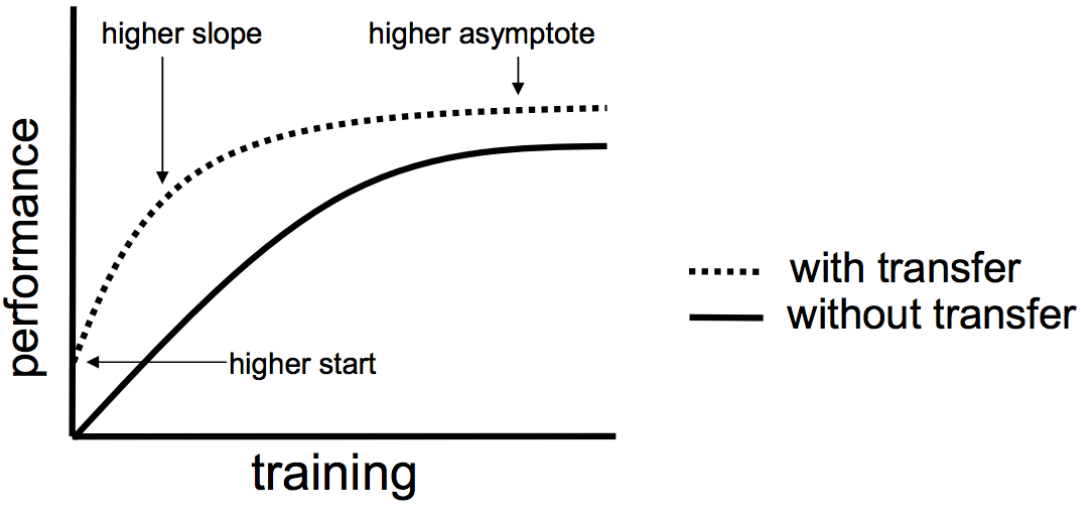


Figure 3.10: Model performance with and without transfer learning

3.6.3.3 Cases to use Transfer Learning

Transfer Learning is used when (a) you don't have enough labeled training data to train your network from scratch and/or (b) there already exists a network that is pre-trained on a similar task, which is usually trained on massive amounts of data. Another case where its use would be appropriate is when Task-a "the task we are dealing with" and Task-b "the that the model is pretrained on" have the same input.

If the original model was trained using TensorFlow, you can simply restore it and re-train some layers for your task.

"Transfer Learning only works if the features learned from the first task are general"

In practice, Transfer Learning from a pretrained ConvNet is typically used in these two different ways [63]:

- Fixed Feature Extraction. Use the output of one intermediate layer to train a machine learning classifier.
- Fine-tuning. Replace the last fully connected layer of the network with a new classification task and use the training data from the new domain to update the weights of some layers. In general, the earlier layers of the ConvNets are not updated and only the deeper ones are updated to the new task.

These two techniques are explored in this work and can be better understood in Chapter 4 - Methodology, Section 4.3 .

3.6.3.4 ConvNets for Image Classification

The convolutional neural networks have a high potential for image classification a general CNN structure (Figure 2.3) is made up of repetitive patterns (which explains the expression deep learning) of Convolutional, ReLU and Pooling layers (considered hidden layers) and finally the fully-connected layers. The resulting volume structure is called feature map (in the case of images, it has a two dimension volume). The learning process (also referred to network training) where weights are optimized is achieved through backpropagation [71].

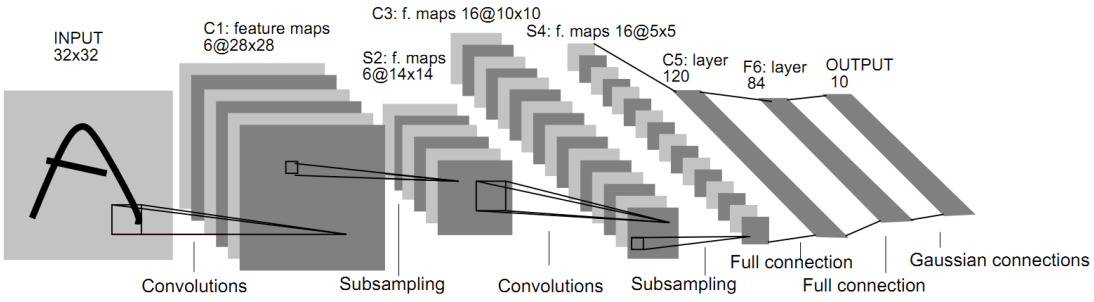


Figure 3.11: LeNet-5 [71] CNN architecture for handwritten digits recognition

3.6.3.5 ConvNets for Image Classification of Skin Lesions

Convolutional neural networks have produced promising results when classifying skin lesions. Examples of related work using deep learning include:

The work of Kawahara et al [62]. explores the idea of using a pretrained ConvNet as a feature extractor to distinguish among 10 classes of non-dermoscopic skin images.

Liao [65] describes an attempt to construct a universal skin disease classification by applying transfer learning on a deep CNN and fine-tuned its weights.

Codella et al. [72] report state-of-the-art performance results using ConvNets to extract image descriptors by using a pre-trained model from the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [70].

Esteva A et al. [73] Where they fine-tuned Google's Inception v3 CNN architecture pretrained on multi source datasets of skin lesion.

Chapter 4

4 Methodology and Results

Introduction

In this chapter, we will present the model architectures for the classification task for the eight lesions and the evaluation, At the beginning of the search for a good ConvNet, we tested three existing models, with the method of transfer learning. The transfer learning is the transfer of weights from a previously trained model. The ConvNets from which we transfer the weights, are trained for huge object classification problems. One of the huge classifications problems was the ImageNet challenge. We fine-tuned these models for our problem and the results where promising.

4.1 Objective

This work develops a study in the Deep Learning field about the impact of generate synthetic samples by dividing original images then combine the resulting slices after being shuffled, and training a models on 2 classes training samples then combine them to get a deep model that can classifier all the eight (8) classes, as a way to defeat the problem of unbalanced data sets that skin lesions dataset from the ISIC archive is known with.

In order to achieve the project goal, 4 successful and well-known Convolutional Neural Networks architectures in the image classification tasks have been adopted VGG16, Inception_v3, DenseNet210 and MobileNet to use it in the web app plus baseline model. The order of execution will be:

Skin lesion classification. The task will perform the automatic classification as one of eight types of lesion. In order to find the best classification model, this task will be divided into three subtasks:

1. **Balanced lesion classification with augmented images.** The four (4) models will perform the classification over the augmented skin RGB images contained in the ISIC dataset the process will have the operations as in section 4.2.6 Data Augmentation .

2. **Balanced lesion classification with synthetic images.** The VGG16 model will perform the classification over the synthetic skin RGB images created from the ISIC dataset the process will have the operations with the same hyperparameters to each experiment in the first task:
    - Resizing to 448 width and high so after dividing the image into 4 slices their dimensions will be 224 to be compatible with pretrained model.
    - Slicing into 4 pieces
    - Shuffling the slices so when combine them we got a different image then the original one.
    - Combining the resulting images with respecting the dimensions for one slice (224) + the original images + the slices.

The objective is to evaluate this approach in its ability to maintain the relevant information in images.. A summary representation of input images to be classified is shown in (Figure 4.1).

To achieve this objective three experiments were applied, first I fine-tuned the VGG16 model and retrained it on sliced data only, second experiment was on combined images only (synthetic), third experiment was on combined images + the original images + the slices, the three experiments used an original images as a validation data set.

3. **Model averaging**
   Model averaging is an ensemble technique where multiple sub-models contribute equally to a combined prediction.
   - Train models with different hyperparameters and architectures.
   - load individual models.
   - perform prediction for each model.
   - average the predictions.

4. **Downgrade classification.** The VGG16 model will perform the classification over one of the best method above, the process will have the bellow operations with the same hyperparameters to each experiment:
   - Train four (4) models each one have 2 different active classes and six (6) not-active (have one Image).
   - Ensemble the models by averaging their weights (Figure 4.2). then retrained the new model, this approach aim to boost information extracting from the images.

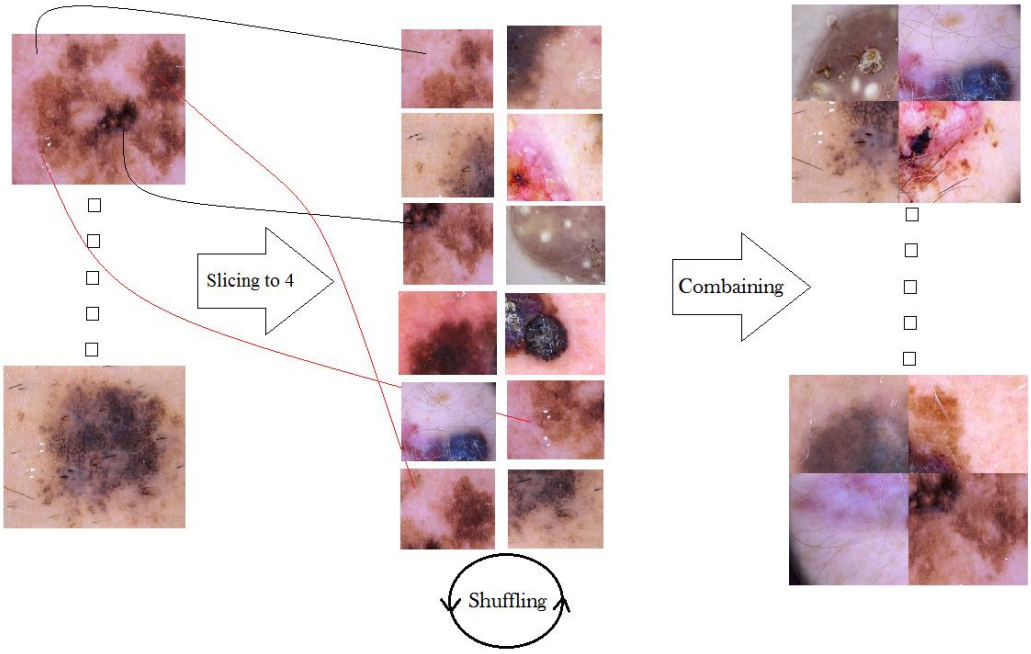


Figure 4.1: Architecture of synthetic samples generation by the proposed approach

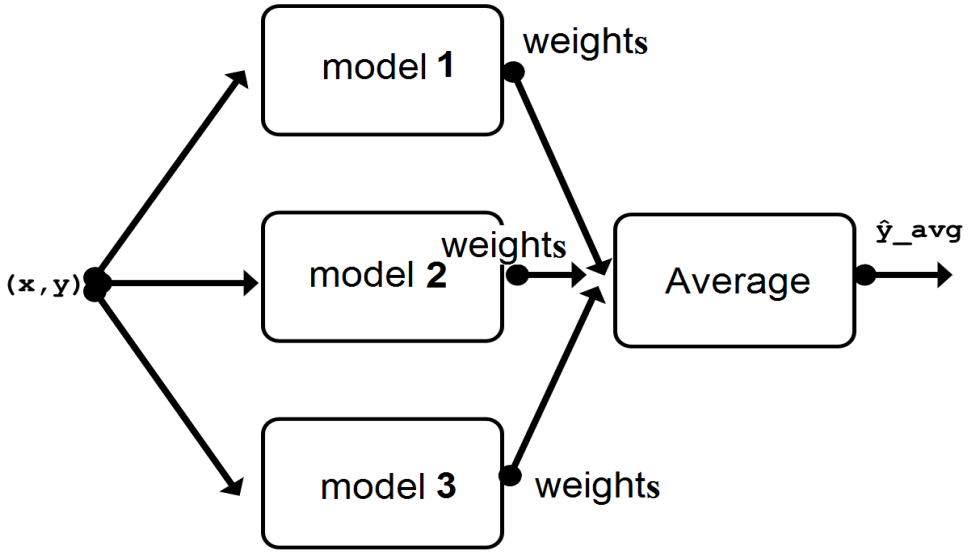


Figure 4.2: Architecture of models ensemble by averaging their predictions

A properly evaluation under the same conditions will be performed over the four (4) classification models and will confirm or refuse the hypothesizes of this work.

4.2 Datasets

The ISBI 2019 Challenge dataset[7] for Skin Lesion Analysis towards melanoma detection was used for this work. The dataset is publicly available and contains 25,332 RGB images. All images are labeled with one type of skin lesion (Table 4.1).

Types	NV	MEL	BKL	BCC	SCC	VASC	DF	AK	Total subset
subset	12875	4522	2624	3323	628	253	239	867	25,331

Table 4.1: ISIC Dataset 2019 [7] distribution

4.2.1 Datasets obstacles

The imbalance between classes is apparent especially for Dermatofibroma and Vascular Lesions. Most of the lesions are belonging to Melanocytic nevus class. (see Table 4.1)

4.2.1.1 The problem of unbalanced classes

In a classification problem when out of all the classes which you want to predict if for one or more classes there are extremely low number of samples you may be facing a problem of unbalanced classes in your data.

The unbalanced classes create a problem due to two main reasons:

- We don't get optimized results for the class which is unbalanced in real time as the model/algorithm never gets sufficient look at the underlying class.
- It creates a problem of making a validation or test sample as its difficult to have representation across classes in case number of observation for few classes is extremely less.

There are three main approaches which are suggested each having its pros and cons:

1. Collect more data

If possible, you could collect more data for the underrepresented classes to match the number of samples in the overrepresented classes. This is probably the most rewarding approach, but it is also the hardest and most time-consuming, if not downright impossible. In the cancer example, there is a good reason that we have way more non-cancer samples than cancer samples: These are easier to obtain, since there are more people in the world who haven't developed cancer.

2. Create copies of training samples

Artificially increase the number of training samples for the underrepresented classes by creating copies. While this is the easiest solution, it wastes time and computing resources. In the cancer example, we would almost have to double the size of the dataset in order to achieve a 50:50 share between the classes, which also doubles training time without adding any new information.

3. Create augmented copies of training samples

Similar to 2, but create augmented copies of the underrepresented classes. For example, in the case of images, create slightly rotated, shifted or flipped versions of the original images. This has the positive side-effect of making the model more robust to unseen examples. However, it only does so for the underrepresented classes. Ideally, you would want to do this for all classes, but then the classes are unbalanced again and we're back where we started.

4. Under-sampling

The method 3 where we augment copies of training samples called Over-sampling, the opposite of it is under-sampling it means we will select only some of the data from the majority class, only using as many examples as the minority class has. This selection should be done to maintain the probability distribution of the class. This two methods are used frequently to solve class-imbalance learning.

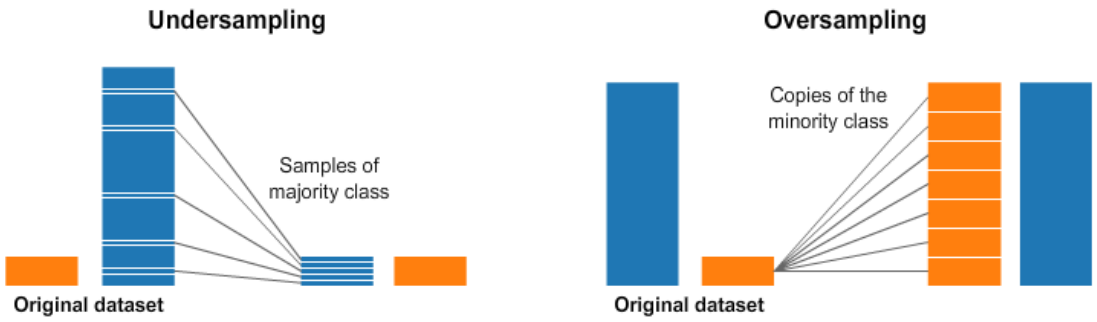


Figure 4.3: Under and Over Sampling

5. Train for sensitivity and specificity

The sensitivity tells us the probability that we detect cancer, given that the patient really has cancer. It is thus a measure of how good we are at correctly diagnosing people who have cancer.

$$\textbf{sensitivity} = p_r\left(\frac{\textbf{detectcancer}}{\textbf{cancer}}\right) = \frac{\boldsymbol{true\ positives}}{\boldsymbol{positives}}$$

The specificity tells us the probability that we do not detect cancer, given that the patient doesn't have cancer. It measures how good we are at not causing people to believe that they have cancer if in fact they do not.

$$\textbf{specificity} = p_r\left(\frac{\rceil \textbf{detectcancer}}{\rceil \textbf{cancer}}\right) = \frac{\boldsymbol{true\ negatives}}{\boldsymbol{negatives}}$$

6. Class-weighted cross-entropy loss

Class-weighted cross-entropy loss are selected as the loss function so as to punish harder on the false predictions on those classes with smaller datasets. As a result, the classification report has improved. The confusion matrix datasets further proves that the model could pay more attention to categories with smaller dataset than normal model after using class weighted loss function.

In this project we used 3, 4, 5 and 6 techniques to solve the problem.

4.2.2 Training Dataset: The sample of data used to fit the model.

The actual dataset that we use to train the model (weights and biases in the case of Neural Network). The model sees and learns from this data.

4.2.3 Validation Dataset:

The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.

The validation set is used to evaluate a given model, but this is for frequent evaluation. We as machine learning engineers use this data to fine-tune the model hyperparameters. Hence the model occasionally sees this data, but never does it "Learn" from this. We(mostly humans, at-least as of 2019) use the validation set results and update higher level hyperparameters. So the validation set in a way affects a model, but indirectly.

4.2.4 Test Dataset:

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained(using the train and validation sets). The test set is generally what is used to evaluate competing models (For example on many Kaggle competitions, the validation set is released initially along with the training set and the actual test set is only released when the competition is about to close, and it is the result of the model on the Test set that decides the winner). Many times the validation set is used as the test set, but it is not good practice. The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world.

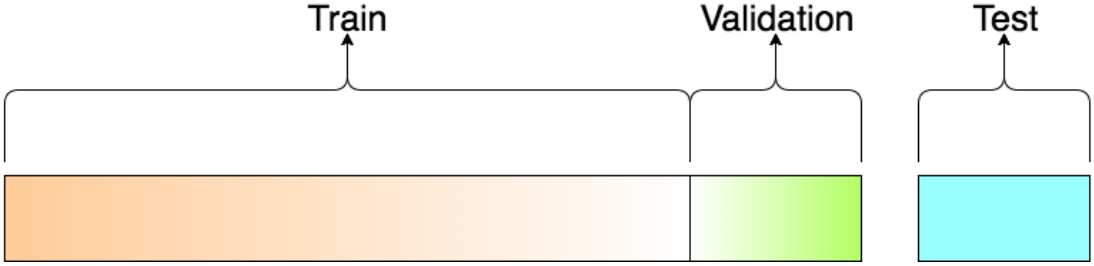


Figure 4.4: Illustration of the Train, Validation and Test datasets

the ISIC archive promised to release the test data in august 2sd, for that the data set were splitted into training and validation data with 20% as validation dataset.

Types	NV	MEL	BKL	BCC	SCC	VASC	DF	AK	Total subset
subset	12875	4522	2624	3323	628	253	239	867	25,331
Training	10300	3618	2099	2658	502	202	191	694	20264
Validation	2575	904	525	665	126	51	48	173	5067

Table 4.2: Dataset after being splitted

4.2.5 Preprocessing

This project takes advantage of ConvNets properties regarding input preprocessing: few previous processing techniques are needed. Although some basic preprocessing forms are performed:

Mean subtraction. In order to center the cloud of RGB values from input data around zero along every dimension of the image, a mean subtraction is applied across the image features.

Image normalization. By dividing each RGB dimension of input images by its standard deviation, a normalization is obtained from its original 0 and 255 pixel values to 1 an 0 normalized values. This preprocessing technique will avoid further issues caused by poor contrast images.

Image cropping & resizing. Input images are preprocessed to be accepted by the architecture though cropping the image to the same aspect ratio as needed and resizing the original image to 224→224 pixels for all the models.

Color Constancy

Robustness is one of the most important characteristics of computer-aided diagnosis systems designed for dermoscopy images. However, it is difficult to ensure this characteristic if the systems operate with multisource images acquired under different setups. Changes in the illumination and acquisition devices alter the color of images and often reduce the performance of the systems. Thus, it is important to normalize the colors of dermoscopy images before training and testing any system[80].

In my project in order to eliminate the variance of luminance and color, we used color constancy with white_patch_retinex algorithm to normalize original images. The difference between original image and processed image by using color constancy algorithm are shown in Figure 3.4.

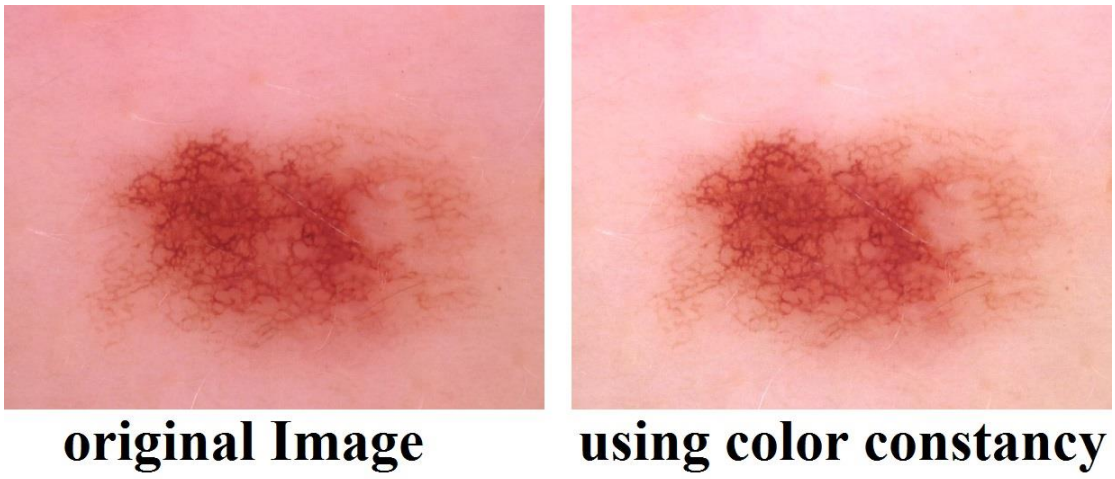


Figure 3.4: Training image with and without using color constancy

4.2.6 Data Augmentation

In order to make the most of our few training examples and increase the accuracy of the model, the ISIC dataset was augmented via a number of random transformations. The selected data augmentation techniques were: size re-scaling, rotations of 180 degrees, horizontal and vertical shift, image zooming, and horizontal and vertical flipping. Furthermore, it is expected that data augmentation should also help prevent overfitting (a common problem in machine learning related to small datasets, when the model, exposed to too few examples, learns patterns that do not generalize to new data) and, for this reason, improving the model's ability to generalize.

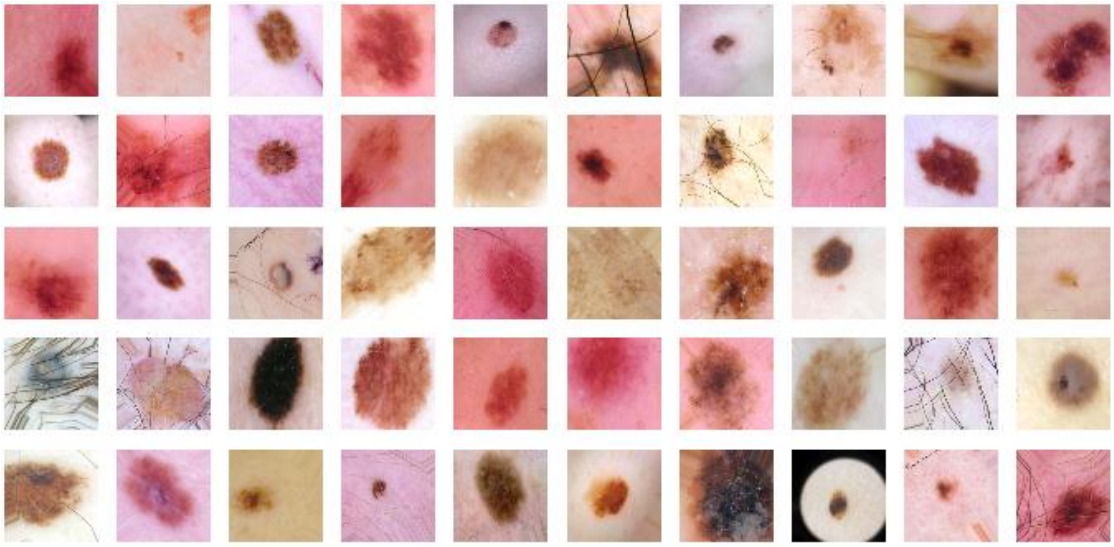


Figure 3.4: Data augmented preview

4.3 Neural Network Architectures used in the experiments

Nowadays, there are several convolutional neural networks architectures that achieved successful results on benchmark challenges such as the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) [48], and as a consequence became very popular in the deep learning community.

In particular, the VGG-16 was in charge of the classification tasks among popular architectures such as the AlexNet [22] and the GoogLeNet [53].

In addition to VGG-16 : Baseline Model, Inception_v3, DenseNet210 and MobileNet will be used.

4.3.1 Baseline Model

Before fine-tuning DCNNs, we build a small CNN to estimate the difficulty of classifying skin lesions. The architecture of the CNN is as follow:

First: A convolutional layer with 16 kernels, each of size 3 and padding such that the size of the image is maintained.

Second: A max pooling layer with window 2x2. The output is feature maps with spatial activation size reduced by a factor of 2.

Third: A convolutional layer with 32 kernels, each of size 3 and padding to maintain the same size.

Fourth: A max pooling layer with window 2x2. The output is feature maps with spatial activation size reduced by a factor of 2.

Fifth: A convolutional layer with 64 kernels, each of size 3 and padding to maintain the same size.

Sixth: A max pooling layer with window 2x2. The output is feature maps with spatial activation size reduced by a factor of 2.

The architecture of this model is heuristically based. we follow the convention in famous DCNNs: using the smallest (3x3) convolutional layers; and double the number of filters in the output whenever the spatial activation size is halved to maintain roughly constant hidden dimensions. To train this model, data augmentation is employed. The intuition of this method is to transform the training dataset a bit in each epoch to produce variation and to guarantee that the model will never see the same image twice. Learning rate is initialized at 0.01 and Adam optimizer is used. Learning rate decay is also used so that the learning rate will halve whenever the validation accuracy plateaus for 3 epochs. Baseline model is trained for a total of 35 epochs.

4.3.2 VGG16

Even though there are many DCNNs model achieving better result on ImageNet than VGG16, we choose to fine-tune VGG16 given its simplicity. (Figure 4.5) is the schema for VGG16. The best performing VGG16 net is similar except that the third, fourth and fifth convolutional block has 4 convolutional layers. VGG16 has 0.901 accuracy for top-5 and 0.713 for top-1 on ImageNet.

Figure 4.5:VGG-16 architecture

To fine-tune VGG16, the top fully-connected layers are removed, and new fully-connected layers (consisting of: one global max pooling layers, one fully connected layer with 512 units, one dropout layer with 0.5 rate, one softmax activation layer for eight (8) types of skin lesions) for our classification tasks are added. Freeze the first convolutional blocks and retrain the last 15 convolutional block of VGG16 and start fine-tune the model for 30 epochs. Throughout the training process, learning rate of 0.001 and Adam optimizer are used. The same data augmentation and learning rate decay strategy as in baseline model is used.

4.3.3 Inception V3

Inception V3 was the top performers on ImageNet with 0.937 accuracy for top-5 and 0.779 for top-1. The namesake of Inception v3 is the Inception modules it uses, which are basically mini models inside the bigger model. The inspiration comes from the idea that you need to make a decision as to what type of convolution you want to make at each layer: Do you want a 3×3? Or a 5×5? The idea is that you don't need to know ahead of time if it was better to do, for example, a 3×3 then a 5×5. Instead, just do all the convolutions and let the model pick what's best. Additionally, this architecture allows the model to recover both local feature via smaller convolutions and high abstracted features with larger convolutions. The larger convolutions are more computationally expensive, so [4] suggests first doing a 1×1 convolution reducing the dimensionality of its feature map, passing the resulting feature map through a ReLU, and then doing the larger convolution (in this case, 5×5 or 3×3). The 1×1 convolution is key because it will be used to reduce the dimensionality of its feature map.

Given Inception V3 trained on ImageNet with 11 inception blocks, we fine-tune the last layer in the pretrained model for 30 epochs.

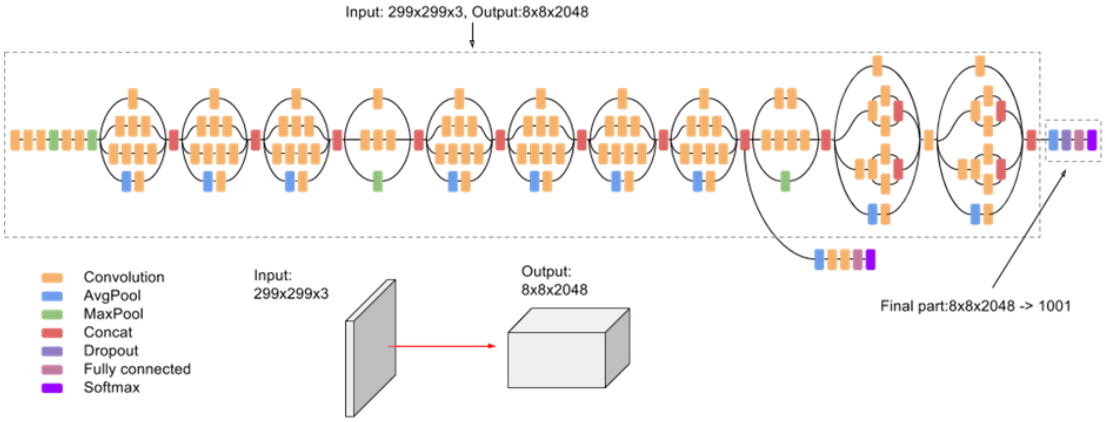


Figure 4.5: Inception architecture V3 with 11 inception blocks

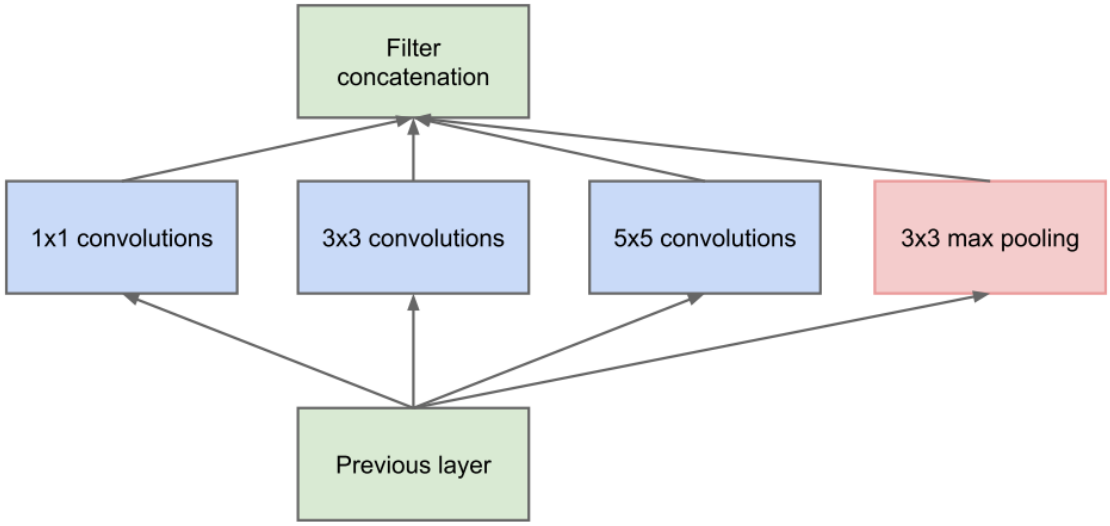


Figure 4.6: Inception Module

4.3.4 Dense Net

Dense Net is a new DCNN architecture introduced that is one of the top performers on ImageNet, 0.936 top-1 and 0.773 top-5. The performance of Dense Net is competitive to Inception V3, but Dense Net has less parameters (approximately 20 million compare with approximate 23 million of Inception V3). Dense Net 201 has 4 dense blocks, Figure 8 displays the general architecture of a dense block. In a dense block, the l th layer has l inputs, consisting of the feature-maps of all preceding convolutional blocks, and its own feature-maps are passed on to all subsequent layers L – l. Each layer reads the state from its preceding layers and writes to the subsequent layer. It changes the state but also passes on information that needs to be preserved. Dense Net architecture explicitly differentiates between information that is added to the network and information that is preserved by concatenating features instead of summing features as in ResNet.

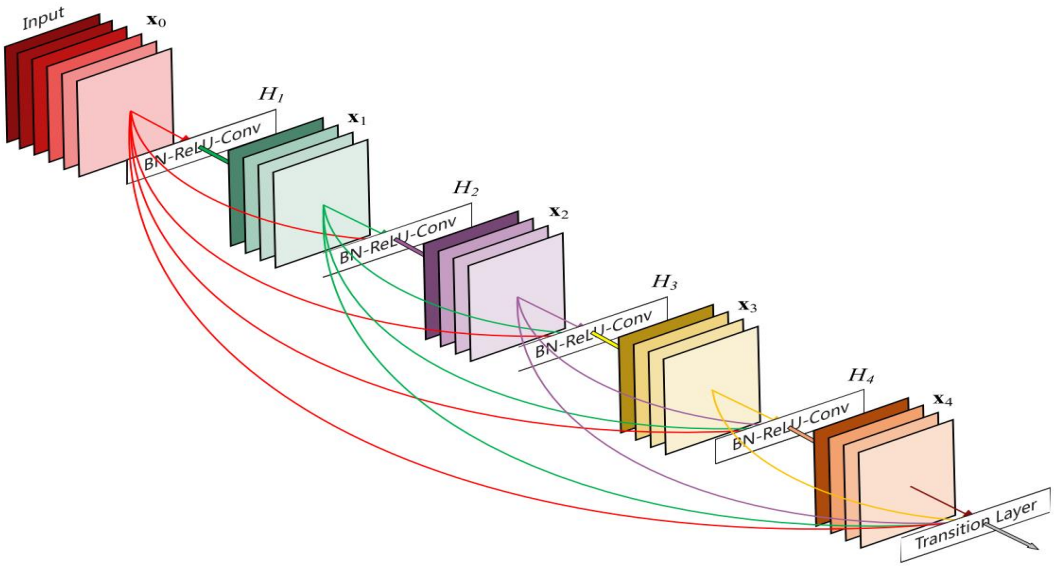


Figure 4.7: A 5-layer Dense Net block

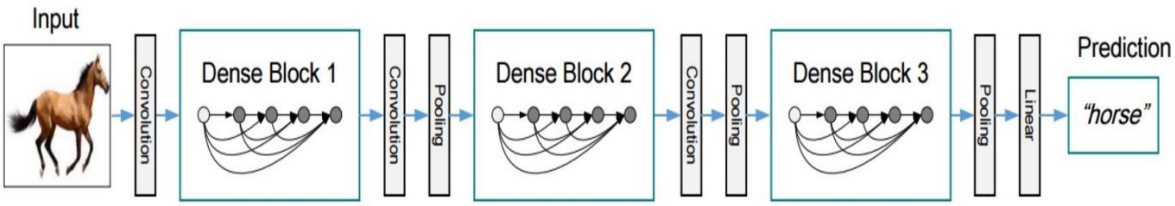


Figure 4.8: A Dense Net with 3 Dense Blocks

In a dense block, one layer generate feature maps through a composite function, consisting of three consecutive operations: batch normalization, ReLU, and a 3x3 convolution. Convolution and Pooling Layers in the between of each dense block is called together a transition layer. Dense Net 201 consists of 4 dense blocks, and we will fine-tune on the last dense block (the last dense block has 32 layers); 2. The same training strategy as in previous sections is used to fine-tune Dense Net. Top layers of DenseNet 201 was fine-tuned for a total of 30 epochs.

we are preforming 30 epochs as constant to all the model as it is enough for most training if the model go overfitting for two much training, we are using a parameter to save only the model which perform better on the validation data set, so the training maybe stop earlier.

4.3.5 Mobilenet

Mobilenet as it is lightweight in its architecture. It uses depthwise separable convolutions which basically means it performs a single convolution on each color channel rather than combining all three and flattening it. This has the effect of filtering the input channels. Or as the authors of the paper explain clearly: " For MobileNet the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size. "

the overall architecture of the Mobilenet is as follows, having 30 layers with :

1. convolutional layer with stride 2
2. depthwise layer
3. pointwise layer that doubles the number of channels
4. depthwise layer with stride 2
5. pointwise layer that doubles the number of channels

   etc.

It is also very low maintenance thus performing quite well with high speed. There are also many flavors of pre-trained models with the size of the network in memory and on disk being proportional to the number of parameters being used.

To fine-tune Mobilenet, the top 5 layers are removed, and new dropout layer with 0.25 rate, one softmax activation layer for eight (8) types of skin lesions) for our classification tasks are added. Freeze the first convolutional blocks and retrain the last 23 layers fine-tune the model for 30 epochs. Throughout the training process, learning rate of 0.001 and Adam optimizer are used. The same data augmentation and learning rate decay strategy as in baseline model is used.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 4.9: MobileNet architecture

5 Requirements and specifications

5.1 ISBI Challenge

The existence of the IEEE International Symposium on Biomedical Imaging (ISBI) [12] 2018 and 2019 challenge for Skin Lesion Analysis Towards Melanoma Detection [13] has provided this project with a useful dataset, the ISIC Archive dataset [7], which contains dermoscopic images paired with their corresponding lesion type.

5.2 Framework used

Python [14] was the main programming language chosen to develop the project.

Keras [15] is a deep learning framework that was initially released on March 2015 [16]. Keras provides a layer of abstraction on top of TensorFlow [17] or Theano [18], which is used as the main neural network framework. Keras is compatible with Python 2.7 and 3.5 and up. It allows for: (1) modularity: users can define their neural networks following a sequence which is a linear stack of layers; (2) minimalism: functions included in the library allow the user to create and modify network layers easily; and (3) extensibility: daily updates provide solutions to ongoing challenges faced by deep learning researchers. Moreover, Keras works on a Python environment, which gives users the freedom to use additional Python dependencies. For all these reasons, Keras was used to implement the neural network architectures on this project.

TensorFlow.js. is an open-source hardware-accelerated JavaScript library for training and deploying machine learning models. Develop ML in the Browser. Use flexible and intuitive APIs to build models from scratch using the low-level JavaScript linear algebra library or the high-level layers API. TensorFlow.js was used to build the final web app.

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Used as an editor for the code.

5.3 GPU power

In addition to Keras, CUDA libraries [19] for parallel computing and programming model developed by NVIDIA were required to utilize the GPUs (Graphics Processing Unit). NVidia GeForce GTX 1050 GPUs were used to train and evaluate the implementation, along with the use of Google Colaboratory which is a free cloud service provided by google for research purposes.

6.4 Results

This section shows and analyzes the results obtained following the guidelines proposed in Chapter 4, proper evaluation methods for classification problems are explained and used.

Model Evaluation

To know if the model can be considered a good model, we need to evaluate it from a different perspectives.

Accuracy: it is a description of systematic errors, a measure of statistical bias; low accuracy causes a difference between a result and a "true" value.

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (5:1)$$

Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of a model.

Precision:

$$\text{PRE} = \frac{TP}{(TP+FP)} \quad (5:2)$$

Recall/sensitivity/TPR:

$$\text{TPR} = \frac{TP}{(TP+FN)} \quad (5:3)$$

macro averaging: we average the performances of each individual class:

$$PRE_{macro} = \frac{PRE_1+\cdots+PRE_k}{k} \quad (5:4)$$

micro averaging:

$$PRE_{micro} = \frac{(TP_1+\cdots+TP_k)}{(TP_1+\cdots+TP_k+FP_1+\cdots+FP_k)} \quad (5:5)$$

		Actual class	
		Class1	**Class2**
Predicted class	**Class1**	5 True Positives	2 False Positives
	Class2	3 False Negatives	17 True Negatives

Table 5.1: Confusion matrix for a binary classification

F1 score: (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5:6}$$

Fine-tuning the top layers for 5 different models:

Model	Validation accuracy	Validation Loss	Depth	# Parameters
Baseline Model	74.48%	0.708	14 layers	3,124,839
MobileNet	76.48%	0.69427	88 layers	4,253,864
VGG16	79.82%	0.646671	23 layers	14,980,935
Inception V3	79.935%	0.6691	315 layers	22,855,463
DenseNet 201	85.8%	0.691	711 layers	19,309,127

Table 4.3: Fine-tuning results

Balanced lesion classification with synthetic images:

The model used is VGG16 for each experiment, we train the last 15 layer with batch size 32 for 20 epochs with the same balanced validation data set.

	Data Type	Validation accuracy	Validation Loss
Original images	Original images (Unbalanced data)	0.71	1.84
Augmented images	Augmented Training set (Balanced data)	0.82	1.17
synthetic images	Combined only (Unbalanced data)	0.69	2.67
	Combined + Slices + Original (Unbalanced data)	0.80	1.26

Table 4.4: synthetic images results

As we can see in the result table training the model on synthetic images did better than the original data only but Unfortunately my approach didn't do better than the image augmentation approach, the different was small, and this motivate us to develop this method and look for better solutions to solve unbalanced data sets.

Downgrade classification

Accuracy

Model	Validation accuracy	Validation Loss	Classes
Model-1	0.945	0.114	NV-MEL
Model-2	0.938	0.380	BKL-BCC
Model-3	0.846	0.882	AK-SCC
Model-4	0.781	0.673	VASC-DF
Ensemble Models Four (4)	0.878	0.244	Eight (8) classes
Multi-class DenseNet	0.894	0.354	Eight (8) classes

Table 4.5: Downgrade ensemble models accuracy results

Ensemble Models Four (4)

Confusion Matrix

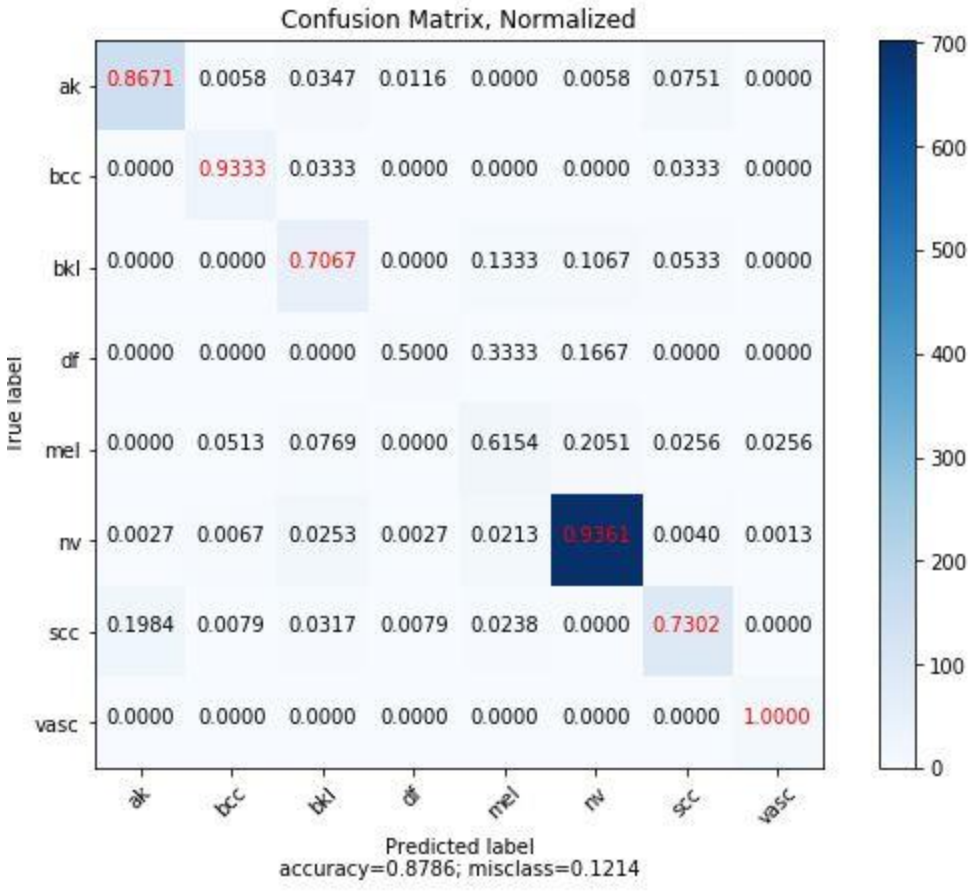


Figure 4.10: ensemble four (4) models confusion matrix

Classification Report

	precision	recall	f1-score	support
ak	0.85	0.87	0.86	173
bcc	0.76	0.93	0.84	30
bkl	0.62	0.71	0.66	75
df	0.38	0.50	0.43	6
mel	0.44	0.62	0.51	39
nv	0.98	0.94	0.96	751
scc	0.81	0.73	0.77	126
vasc	0.85	1.00	0.92	11
accuracy			0.88	1211
macro avg	0.71	0.79	0.74	1211
weighted avg	0.89	0.88	0.88	1211

Figure 4.11: ensemble four (4) models classification report

Model averaging

Accuracy

Model	Validation accuracy	Validation Loss	Classes
Model-dense	0.894	0.354	Eight (8) classes
Model-Inception_V3	0.822	0.293	Eight (8) classes
Model-InceptionResNetV2	0.895	0.563	Eight (8) classes
Ensemble Models dense-Inception_V3	0.881	0.284	Eight (8) classes

Table 4.6: Ensemble models results

1-Model-DenseNet

Confusion Matrix

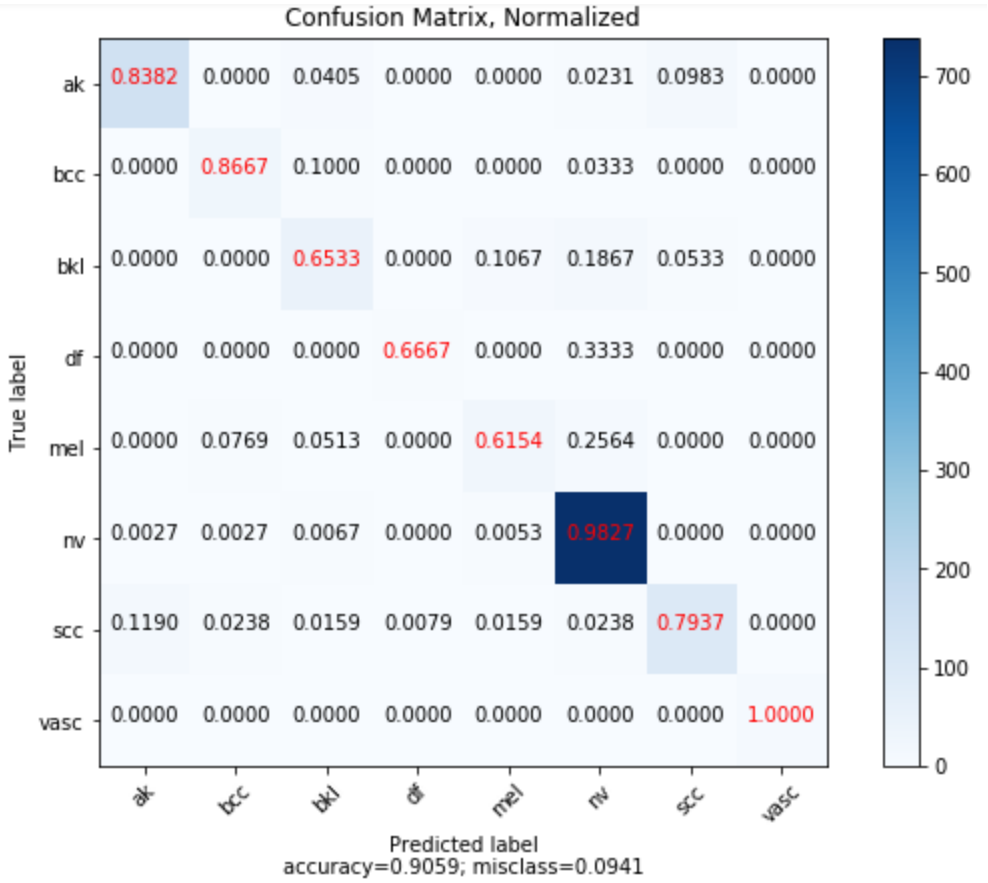


Figure 4.12: Full retrained DenseNet confusion matrix

Classification Report

	precision	recall	f1-score	support
ak	0.90	0.84	0.87	173
bcc	0.76	0.87	0.81	30
bkl	0.72	0.65	0.69	75
df	0.80	0.67	0.73	6
mel	0.63	0.62	0.62	39
nv	0.96	0.98	0.97	751
scc	0.83	0.79	0.81	126
vasc	1.00	1.00	1.00	11
accuracy			0.91	1211
macro avg	0.82	0.80	0.81	1211
weighted avg	0.90	0.91	0.90	1211

Figure 4.13: Full retrained DenseNet classification report

2- Model-Inception_V3

Confusion Matrix

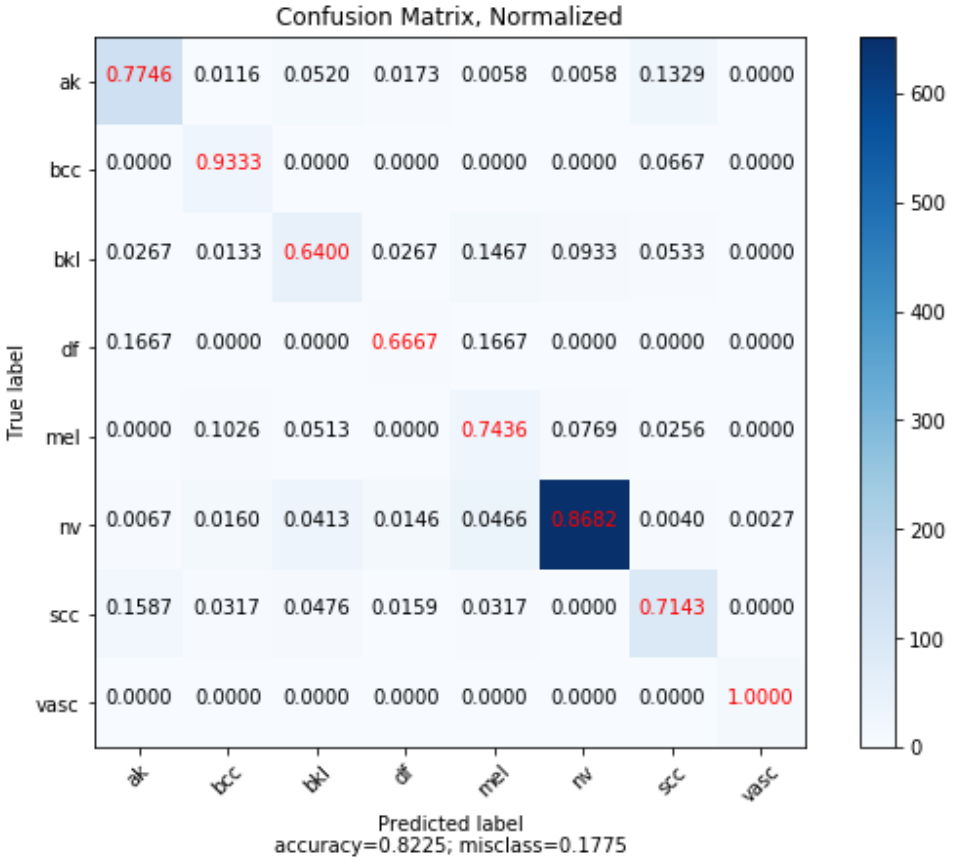


Figure 4.14: Full retrained Inception_V3 confusion matrix

Classification Report

	precision	recall	f1-score	support
ak	0.83	0.77	0.80	173
bcc	0.55	0.93	0.69	30
bkl	0.50	0.64	0.56	75
df	0.18	0.67	0.29	6
mel	0.36	0.74	0.48	39
nv	0.98	0.87	0.92	751
scc	0.73	0.71	0.72	126
vasc	0.85	1.00	0.92	11
accuracy			0.82	1211
macro avg	0.62	0.79	0.67	1211
weighted avg	0.87	0.82	0.84	1211

Figure 4.15: Full retrained Inception_V3 classification report

3- Ensemble Model denseNet-Inception_V3

Confusion Matrix

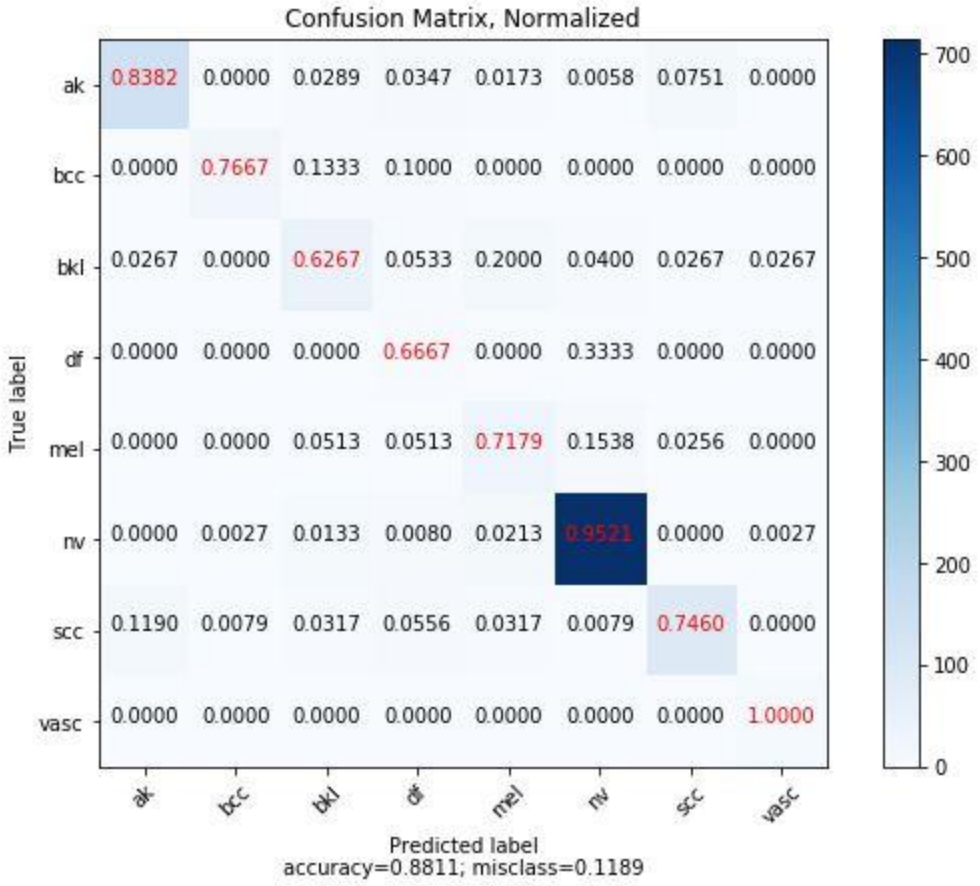


Figure 4.16: Ensemble Model denseNet-Inception_V3 confusion matrix

Classification Report

	precision	recall	f1-score	support
ak	0.90	0.84	0.87	173
bcc	0.88	0.77	0.82	30
bkl	0.65	0.63	0.64	75
df	0.12	0.67	0.21	6
mel	0.42	0.72	0.53	39
nv	0.98	0.95	0.97	751
scc	0.85	0.75	0.80	126
vasc	0.73	1.00	0.85	11
accuracy			0.88	1211
macro avg	0.69	0.79	0.71	1211
weighted avg	0.91	0.88	0.89	1211

Figure 4.17: Ensemble Model denseNet-Inception_V3 classification repport

Chapter 5

5 Conclusion and Future Work

5.1 conclusion

Skin cancer is increasing and affects many people in different countries in the world including Algeria. Malignant melanoma is the deadliest type of skin cancer that can be treated successfully if it is detected early. Early intervention will lead to a better survival rate. Since the clinical observation of a melanoma is subject to human error, early detection can be enhanced by utilizing an automated process. This project has proposed a Deep Learning solution for assisting dermatologists during the diagnosis of skin lesions. More specifically, it has investigated how ensemble of binary models can be averaged to improves the performance of a fine-tuned convolutional neural network model approach for a multi-class classifier for early skin lesion detection.

The proposed approach consist of dividing the multi-class problem (8 classes) to binary classification, each two classes are classified separately, by that will have four (4) trained models specialized in two classes the result of combining this model is one multi-class model, that can classifier all our eight (8) classes, By averaging their weights then retrained it again, this approach aim to boost information extracting from the images to enhance the accuracy of the model.

However, accuracy is not the full story and may not be the best measure of performance in a medical setting. Sensitivity is often considered a more important metric in the medical setting. In these situations, where early diagnosis is of great importance, it is better to raise a false positive than to create a false negative – the project would rather be overly pessimistic than optimistic in its prediction, to implement this a class weighted loss function was used.

The skin lesions dataset was very imbalanced classes, this lead us to think in way to increase images in law classes by generating a synthetic images, but unfortunately my approach was not better than the state of the arte, still it was used in this project.

5.2 Future Work

Further work could explore the performance of applying Squeeze-and-Excitation [79] blocks in the architecture, also Residual Neural Networks , that have recently performed excellent results on image classification tasks by proposing substantially deeper and easier to optimize networks.

Future work aims to implement some techniques that we encountered while reading related papers to reach the goal of this projects. As one of the biggest problems of Convolutional neural networks is the computational power, wherefore the need to approach to drastically reduce the training memory needed for the training, InPlace-ABN [78] can save up to 50% of GPU memory required to train deep neural network models.

The second common problem is data, the CNNs it require a huge amount of training data and, because the ISIC data set is unbalance, we believe applying Deep rule-based (DRB) [77] approach on my classifier will help in the performance.

Furthermore, changing the loss function to Focal loss, to reduce the harm of the unbalance data on the model performance, where its proves to be more stable.

Personal area of interest is lay on the automated healthcare research, where we believe it's worth the time and efforts.

Bibliography

[1] W. Sterry and R. Paus. Thieme clinical companions dermatology. In Thieme Verlag. Stuttgart, New York., 2006.

[2] Yolanda Smith, "Human Skin structure," . [online]. https://fr.scribd.com/document/313500758/Lit-SkinStruct-Bensouillah-Ch01-pdf [Accessed: march 14 2019].

[3] Matthew Hoffman, "Picture of the skin and skin cases," . [online]. http://www.webmd.com/skin-problems-and-treatments/picture-of-the-skin#1 [Accessed: march 14 2019].

[4] Skin Cancer Foundation. Skin cancer facts and statistics. [online]. http://www.skincancer.org/skin-cancer-information/skin-cancer-facts , 2019. [Accessed: march 15 2019].

[5] American Cancer Society. Skin cancer prevention and early detection. [online]. https://www.cancer.org/cancer/skin-cancer/prevention-and-early-detection.html , 2015. [Accessed: march 15 2019].

[6] Jones and Bartlett "Basic Biology of the Skin," . http://samples.jbpub.com/9780763761578/03ch_pg029-032_Wiles.indd.pdf

[7] ISIC Archive. International skin imaging collaboration: Melanoma project website. [online]. https://isic-archive.com , 2019.

[8] J. Kawahara, A. BenTaieb, and G. Hamarneh. Deep features to classify skin lesions. In IEEE International Symposium on Biomedical Imaging (IEEE ISBI), 2016.

[9] Liao, Haofu. "A Deep Learning Approach to Universal Skin Disease Classification." (2015). https://pdfs.semanticscholar.org/af34/fc0aebff011b56ede8f46ca0787cfb1324ac.pdf

[10] N. Codella, Q.B. Nguyen, S. Pankanti, D. Gutman, B. Helba, A. Halpern, and J.R. Smith. Deep learning ensembles for melanoma recognition in dermoscopy images. In arXiv preprint arXiv:1610.04662, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105), 2012.

[12] ISBI. Ieee international symposium on biomedical imaging. [online]. http://biomedicalimaging.org/ , 2019.

[13] International Symposium on Biomedical Imaging. Isbi 2016: Skin lesion analysis towards melanoma detection. [online]. https://challenge2019.isic-archive.com/ , 2019.

[14] Python Software Foundation. Python programming language. [online]. https://www.python.org , 2019.

[15] F. Chollet. Keras documentation. [online]. https://keras.io/ , 2019.

[16] F. Chollet. Introducing keras 1.0. [online]. https://blog.keras.io/introducing-keras-10.html , March 2015.

[17] TensorFlow Development Team. Tensorflow library for machine intelligence. [online]. https://www.tensorflow.org , 2019.

[18] Theano Development Team. Theano python library. [online]. http://deeplearning.net/software/theano/# , 2017.

[19] NVidia. Cuda parallel computing platform. [online]. http://www.nvidia.com/object/cuda_home_new.html , 2019. [Accessed: January 29 2019].

[20] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation, 1(4):541–551, 1989.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3): 211–252, 2015.

[23] Ian Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. CoRR, abs/1312.6082, 2013. URL http://arxiv.org/abs/1312.6082.

[24] Dan Cire¸san, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. Neural Networks, 32:333–338, 2012.

[25] Ronan Collobert and Jason Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008, pages 160–167, 2008.

[26] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Lecun Y., Bengio Y., and Hinton G. Deep learning. Nature, 521(7553):436–444, 2015.

[27] David A. Forsyth and Jean Ponce. Computer Vision - A Modern Approach, Second Edition. Pitman, 2012. ISBN 978-0-273-76414-4.

[28] "Human Skin," Virtual Medical Center, 2005 [online]. https://www.myvmc.com/anatomy/human-skin [Accessed: April 03 2019].

[29] William Montagna, F. John G. Ebling, "Human Skin," Encyclopaedia Britannica, 2017.

[30] Heather Brannon, "Understanding the Epidermis," [online]. https://www.verywell.com/anatomy-of-epidermis-1068881 , 2019 [Accessed: April 05 2019].

[31] Matthew Hoffman, "Picture of the skin and skin cases," [online]. http://www.webmd.com/skin-problems-and-treatments/picture-of-the-skin#1 [Accessed: April 06 2019].

[32] Mayo Clinic Staff "Seborrheic keratosis symptoms and causes", 2018 [online]. https://www.mayoclinic.org/diseases-conditions/seborrheic-keratosis/symptoms-causes/syc-20353878 [Accessed: April 06 2019]

[33] Dr Amanda Oakley, Dermatologist "Dermatofibroma", 2016 [online]. https://www.dermnetnz.org/topics/dermatofibroma/ [Accessed: April 06 2019]

[34] Sardana K, Chakravarty P, Goel K. "Optimal management of common acquired melanocytic nevi (moles): current perspectives". Clin Cosmet Investig Dermatol. 2014;7:89–103. Published 2014 Mar 19. doi:10.2147/CCID.S57782

[35] British Association of Dermatologists "Melanocytic naevi" , 2017 [online]. https://www.bad.org.uk/shared/get-file.ashx?id=100&itemtype=document [Accessed: April 06 2019]

[36] FERN A. WIRTH, M.D., and MARK H. LOWITT, M.D., "Diagnosis and Treatment of Cutaneous Vascular Lesions" 1998 [online]. https://www.aafp.org/afp/1998/0215/p765.html [Accessed: April 06 2019]

[37] Mark Lebwohl, Deborah S. Sarnoff 2019 "Actinic Keratosis" [online]. https://www.skincancer.org/skin-cancer-information/actinic-keratosis [Accessed: April 06 2019]

[38] Mayo Clinic Staff "Basal cell carcinoma" , 2018 [online]. https://www.mayoclinic.org/diseases-conditions/basal-cell-carcinoma/symptoms-causes/syc-20354187 [Accessed: April 06 2019]

[39] American Academy of Dermatology "Basal cell carcinoma" , 2018 [online]. https://www.aad.org/public/diseases/skin-cancer/basal-cell-carcinoma [Accessed: April 06 2019]

[40] Mayo Clinic Staff "Melanoma" , 2019 [online]. https://www.mayoclinic.org/diseases-conditions/melanoma/symptoms-causes/syc-20374884 [Accessed: April 06 2019]

[41] Mayo Clinic Staff "Squamous cell carcinoma of the skin" , 2018 [online]. https://www.mayoclinic.org/diseases-conditions/squamous-cell-carcinoma/symptoms-causes/syc-20352480 [Accessed: April 06 2019]

[42] Aurora Saez, Begona Acha and Carmen Serrano, "Pattern Analysis in Dermoscopic Images," Computer vision techniques for the diagnosis of skin cancer, series in BioEngineering, Springer, pp. 23-47, 2014.

[43] Rodney Sinclair "Skin cancer and benign lesions," 2012.

[44] Wisconsin, M.C.o., "Malignant Melanoma: staging". Collaborative Hypertext of Radiology, 2006.

[45] Balch C, B.A., Soong S, Atkins M, Cascinelli N, Coit D, Fleming I,, Houghton A, Kirkwood J, McMasters K, Mihm M, Morton D, Reintgen D, Ross M, Sober A, Thompson J, Thompson J, "Final version of the American Joint Committee on Cancer staging system for cutaneous melanoma". Gershenwald J and J Clin Oncol, 2001. 19(16): p. 3635–48.

[46] Weedon, D., Skin pathology, 2nd Edition 2002, Sydney: Churchill-Livingstone.

[47] Retsas, S., K. Henry, and M.Q. Mohammed, Prognostic factors of cutaneous melanoma and a new staging system proposed by the American Joint Committee on Cancer (AJCC): validation in a cohort of 1284 patients. Eur J Cancer, 2002. 38: p. 511-516.

[48] Breslow, A., Thickness, cross-sectional areas and depth of invasion in the prognosis of cutaneous melanoma. Ann Surg, 1970. 172: p. 902-908.

[49] Maryam Sadeghi, "Towards Prevention and Early Diagnosis of Skin Cancer: ComputerAided Analysis of Dermoscopy Images," 2012.

[50] Nilkamal S. Ramteke and Shweta V. Jain, "ABCD rule based automatic computer-aided skin cancer detection using MATLAB," Int. J. Computer technology & Applications, Vol. 4, pp. 691-697, 2013.

[51] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. Book in preparation for MIT Press, 2016. URL http://www.deeplearningbook.org.

[52] Thomas M. Mitchell. Machine Learning. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 9780070428072.

[53] Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2. ISBN 978-0-07-042807-2.

[54] Jürgen Schmidhuber. Deep Learning in Neural Networks: An Overview. Neural Networks, 61: 85–117, 2015.

[55] Jürgen Adamy. Fuzzy Logik, Neuronale Netze und Evolutionäre Algorithmen. Shaker Verlag, 2011. ISBN 978-3-8440-0397-0. 3., überarbeitete Ausgabe.

[56] Stuart J. Russell and Peter Norvig. Künstliche Intelligenz. Ein moderner Ansatz. Pearson Studium, 2004. ISBN 9783827370891. 2., überarbeitete Ausgabe.

[57] Michael A. Nielsen. Neural Networks and Deep Learning. Determination Press, 2015. [online]. http://neuralnetworksanddeeplearning.com . [Accessed: February 01 2019].

[58] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 9:249–256, 2010.

[59] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. CoRR, abs/1609.04836, 2016. URL http://arxiv.org/abs/1609.04836.

[60] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, AISTATS, volume 15 of JMLR Proceedings, pages 315–323. JMLR.org, 2011.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. arXiv preprint arXiv:1512.03385, 2015.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. CoRR, abs/1603.05027, 2016. URL http://arxiv.org/abs/1603.05027.

[63] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. CoRR, abs/1512.00567, 2015. URL http://arxiv.org/abs/1512.00567.

[64] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex A. Alemi. Inception-v4, InceptionResNet and the Impact of Residual Connections on Learning. In ICLR 2016 Workshop, 2016. URL https://arxiv.org/abs/1602.07261.

[65] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In David Blei and Francis Bach, editors, Proceedings of the 32nd International Conference on Machine Learning (ICML-15), volume 37, pages 448–456. JMLR Workshop and Conference Proceedings, 2015.

[66] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). CoRR, abs/1511.07289, 2015. URL http://arxiv.org/abs/1511.07289 .

[67] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research (JMLR), 15:1929–1958, 2014.

[68] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann Lecun, and Christoph Bregler. Efficient object localization using Convolutional Networks. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 648–656, June 2015.

[69] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Lecture Notes to CS231n: Convolutional Neural Networks for Visual Recognition, 2015. [online]. https://cs231n.github.io . [Accessed: January 15 2019].

[70] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In ICLR (workshop track), 2015.

[71] Abrar H. Abdulnabi, Gang Wang, Jiwen Lu, and Kui Jia. Multi-Task CNN Model for Attribute Prediction. IEEE Trans. Multimedia, 17(11):1949–1959, 2015.

[72] Peijin Ji, Lianwen Jin, and Xutao Li. Vision-based Vehicle Type Classification Using Partial Gabor Filter Bank. In 2007 IEEE International Conference on Automation and Logistics, pages 1037–1040, 2007.

[73] Esteva A, Kuprel B, Novoa RA, Ko, Justin, Swetter SM, Blau HM, Thrun S. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 115 2017;542(7639)

[74] Gary Marcus. Hyping Artificial Intelligence, Yet Again. December 31, 2013. [online]. https://www.newyorker.com/tech/annals-of-technology/hyping-artificial-intelligence-yet-again [Accessed: March 20 2019].

[75]. Sean Tao , Deep Neural Network Ensembles, **arXiv:1904.05488**, 2019. URL https://arxiv.org/abs/1904.05488 .

[76]. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár Facebook AI Research (FAIR). Focal Loss for Dense Object Detection, **arXiv:1708.02002** . 2017 URL https://arxiv.org/abs/1708.02002

[77]. P. Angelov, Plamen & Gu, Xiaowei. Deep Rule-Based Classifier with Human-level Performance and Characteristics. Information Sciences. 2018. 463-464; 10.1016/j.ins.2018.06.048.

[78]. Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder. In-Place Activated BatchNorm for Memory-Optimized Training of DNNs. **arXiv:1712.02616** **.** 2017 URL https://arxiv.org/abs/1712.02616

[79]. Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu. Squeeze-and-Excitation Networks, **arXiv:1709.01507** **.** 2017 URL https://arxiv.org/abs/1709.01507

[80] Catarina Barata, M Emre Celebi, and Jorge S Marques. Improving dermoscopy image classification using color constancy. IEEE journal of biomedical and health informatics, 19(3):1146–1152 . 2015 URL https://ieeexplore.ieee.org/document/6866131

[81] Lisa Torrey and Jude Shavlik, Handbook of Research on Machine Learning Applications . IGI Global (2009) URL ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf