Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2020

Fábio Santos

# Aprendizagem automática para o diagnostico de lesões de pele (MÁXIMO 70 CARACTERES)

# Machine learning for the diagnosis of skin lesions (MAX 70 CHARACTERS)

# DOCUMENTO PROVISÓRIO

Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2020

Fábio Santos

# Aprendizagem automática para o diagnostico de lesões de pele (MÁXIMO 70 CARACTERES)

# Machine learning for the diagnosis of skin lesions (MAX 70 CHARACTERS)

# DOCUMENTO PROVISÓRIO

"*The greatest challenge to any thinker is stating the problem in a way that will allow a solution*"

— Bertrand Russell

**Universidade de Aveiro**
**2020**

Departamento de Eletrónica,
Telecomunicações e Informática

**Fábio Santos**

# Aprendizagem automática para o diagnostico de lesões de pele (MÁXIMO 70 CARACTERES)

# Machine learning for the diagnosis of skin lesions (MAX 70 CHARACTERS)

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor (nome do orientador), Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor (co-orientador), Professor auxiliar convidado do Departamento de Matemática da Universidade de Aveiro.

Texto Apoio financeiro do POCTI no âmbito do III Quadro Comunitário de Apoio.

Texto Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio.

Dedico este trabalho aos meus pais, amigos, e em especial ao orientador e co-orientadora.

**o júri / the jury**

presidente / president

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

vogais / examiners committee

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

**agradecimentos / acknowledgements**

Agradeço toda a ajuda a todos os meus colegas e companheiros.

**Palavras Chave**

texto livro, arquitetura, história, construção, materiais de construção, saber tradicional.

**Resumo**

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

**Keywords** textbook, architecture, history, construction, construction materials, traditional knowledge.

**Abstract**

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **ANN** | Artificial Neural Network |
| **CNN** | Convolutional Neural Network |
| **GPU** | Graphics Processing Unit |
| **CPU** | Central Processing Unit |
| **RAM** | Random Access Memory |
| **ISIC** | International Skin Imaging Collaboration |
| **ILSVRC** | ImageNet Large Scale Visual Recognition Competition |
| **HAM10000** | Human Against Machine with 10000 images |

CHAPTER 1

# Introduction

*A sort description of the chapter.*
*A memorable quote can also be used.*

## 1.1 Background

Skin cancer is the most common type of cancer, particularly, in the United States of America the incidence rates keep rising with currently 1 in 5 persons developing skin cancer until the age of 70 [1]. However, skin cancer represents a problem not only for America but also for the international health community in general. For example, in Europe, over 100000 people are diagnosed with melanoma and 22000 deaths annually occur due to this form of skin cancer [2]. Yet, one of the most remarkable facts about skin cancer is that when detected on late stages there is a 23% chance of survival, but when detected early the 5 year survival rate rises to 99% [1]. Therefore, the early detection of skin cancer is an absolute priority.

Skin cancer can be detected by dermatology professionals by simple visual examination of skin lesions. However, the difference between malignant and benign skin lesions can be negligible making it a difficult task even for trained medical experts. As such, a medical application which provides automated skin lesion diagnosis for decision support is an welcome addition to this field.

Initially, automated diagnosis of skin lesions was made based on predefined techniques well known by dermatology professionals such as the ABCDE rule (Asymmetry, Border, Color, Dermoscopic structure and Evolving), but often failed to either generalize to new cases or lacked the accuracy of a human. However, in more recent years, machine learning approaches into skin lesion diagnosis shows remarkable performance in comparison with the hand crafted algorithms, specially with deep learning methods[3][4].

## 1.2 Objectives and Motivation

A strong assumption can be made that if an accurate automated skin lesion diagnosis tool is used by dermatologists, then skin cancer cases will be detected earlier. Therefore, the

main objective behind this dissertation is to improve the current work on automated skin lesion diagnosis by using deep learning techniques. This work will be part of a tool that has the intent of being used in a clinic context, so the priority is to its performance as much as possible.

Finally, the aforementioned tool must be packaged within a eHealth application, in order to be easily accessed by medical professionals in a clinical environment. Such application must have the ability to potentially integrate other components useful for both dermatologists and patients while enabling easy communication between them.

## 1.3 Outline

This dissertation is organized into four other chapters

- Chapter 2 introduces the reader to deep learning concepts and techniques relevant to the current state-of-the-art and reviews work related to this dissertation;
- Chapter 3 introduces the hardware, software, and dataset used for this work;
- Chapter 4 thoroughly describes the experiments and presents results;
- Chapter 5 offers final remarks, key takeaways, and directions for future work;

CHAPTER 2

# State of the art

The following sections are structured as follows. Section 2.1 provides an exploratory look into eHealth/mHealth applications for skin lesion diagnosis. Section 2.2 gives an overview of deep learning applied into medical imaging. Finally, Section 2.3 reviews some of the current approaches towards skin lesion diagnosis systems using deep learning.

## 2.1 Methods and materials

The following sections are structured as follows. Section 3.1 provides an explanatory view into the most used image recognition neural network typology. Section 3.2 describes two of the most popular Convolutional Neural Network (CNN) architectures. Section 3.3 explores the concept of repurposing pre-trained models from CNN architectures trained on generic datasets. Section 3.4 explores problems related to training deep networks and some solutions. Finally, Section 3.5 explores state of the art frameworks for training and deploying deep networks.

### 2.1.1 Artificial Neural Networks

Artificial Neural Networks (Artificial Neural Network (ANN)s) compose a category of machine learning algorithms that are inspired by biological neural networks. These structures are comprised by multiple layers, each composed by multiple neurons that can be interpreted as a function described by parameters. Within a neuron, each input $x_i$ is multiplied by a learnable matrix of weights $w_i$, where each weight can be seen as the synaptic strength between two neurons. A second parameter is also taken into consideration called bias $b$ that is added to the element wise multiplication between the weights and inputs matrices. With this two parameters a neuron will output a signal according to a activation function $f$ which introduces non-linearity. Mathematically speaking, the output of a neuron can be expressed as Equation 2.1.

$$f(\sum_{i}^{n} x_i w_i + b) \tag{2.1}$$

Some activation functions that can be used are:

- The sigmoid function $f(x) = (\frac{1}{1+exp^{-x}})$. Provides smooth gradient between 0 and 1 values, but suffers from computation issues (vanishing gradients problem).
- The tanh function $f(x) = (\frac{e^x - e^{-x}}{e^x + e^{-x}})$. Like the sigmoid except that output values range from -1 to 1, meaning it's center is at 0.
- The ReLU function $f(x) = max(0, x)$. A network with such neurons is able to overcome numerical computation issues (exploding and vanishing gradients, typically associated with activation functions like the signmoid function) which, in practice, allow faster convergence **?**.
- The softmax function $f(x_i) = \frac{exp(x_i)}{\sum_i exp(x_j)}$. It is typically used in the output layer for neural networks which classify inputs into multiple classes. It normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class **?**.

We can look at ANNs as an iterative process that tries to optimize parameters in order to minimize a cost function.

ANNs can approximate any function with just one layer but for more complex functions an increased hidden width may be required which cloud limit it's capability **?**. Therefore, networks with many more layers and with an organization which allows them to create levels of abstraction are often used to solve more complex problems. We call such networks deep neural networks.

There are many different typologies of ANNs. The most common has a fully connected structure between layers, where each neuron is connected to every other neuron in the previous layer.

However, in image recognition such network architecture does not take into account the spatial structure of images. For example, it treats input pixels which are far apart and close together on exactly the same footing.

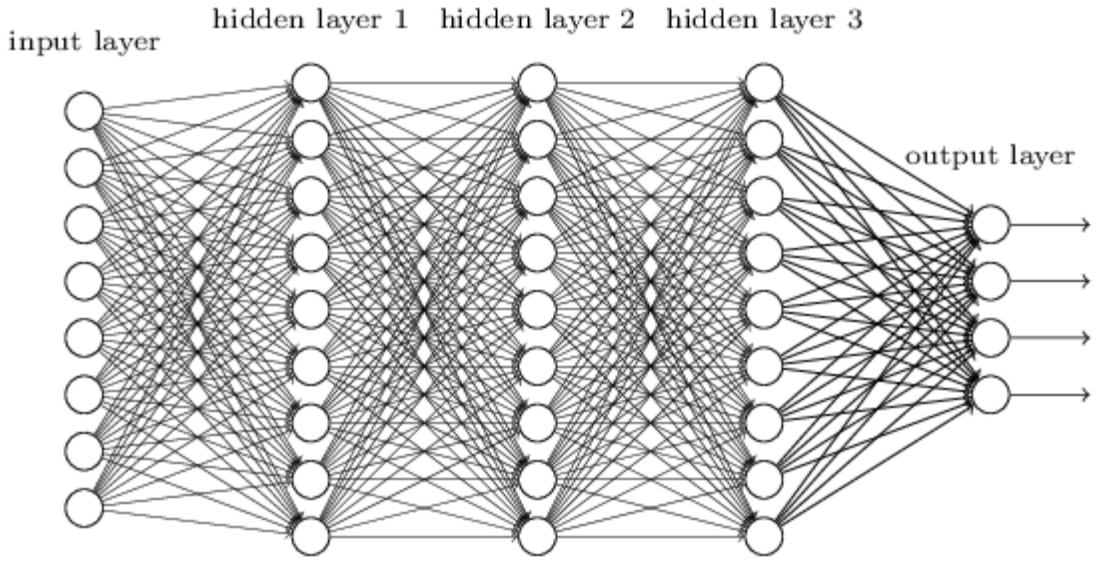


**Figure 2.1:** Artificial neural network [5]

### 2.1.2 Convolutional Neural Networks

Instead, convolutional neural networks (CNN) or some close variant are used in most neural networks for image recognition problems[6]. They still retain the core concepts of ANNs, but add 3 different concepts which distinguish them from conventional ANNs:

- Local receptive fields: In Figure 2.1 inputs were depicted as a vertical line of neurons that were fully connected to the next hidden layer. However, for images inputs are pixel intensities in 2D space. Convolutional layers exploit this concept by only connecting neurons to a particular region of the input volume which ensures that the learnt filters activate strongly only in the presence of a spatially local input pattern. That region of the input is called local receptive field and is usually characterized by it's square size (5x5 for example) and it's stride length which can be 1 or more. Note that if we consider 5 x 5 local receptive field with stride 1 and a 28 x 28 input image then there will be 24 x 24 neurons in the hidden layer, because it can only move 23 neurons across.
- Shared weights: Weights and biases are shared across the hidden neurons so that convolutional networks become well adapted to translation variances in images. The shared weights are often said to define a kernel or filter, while to the map from the input layer to the hidden layer we call feature map, where a feature detected by a hidden neuron is some kind of input pattern that will cause the neuron to activate. To do image recognition we need multiple feature maps in order to recognize multiple features.
- Pooling layers: These layers simplify the information in the output from the convolutional layer by removing unnecessary information, such as noise. A common pool layer is max pooling which provides a way to know if a given feature is found anywhere in a region of a image. For each feature map, it works by sliding a filter of size l x l with a specific stride S and computing the maximum value for the selected part of the input (illustrated in figure 2.2). Overall this reduces the number of free parameters (while introducing no new parameters since max pooling is a fixed function of the input) and the memory footprint and computation in the network [6].

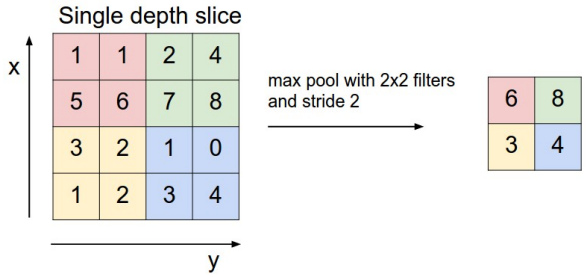


**Figure 2.2:** Max pooling with a 2 × 2 filter and stride S = 2 [6]

As a result of these 3 concepts the overall architecture of convolutional neural networks becomes quite different from fully connected neural networks but the same basic concepts still applies. Namely, the main objective is still to train the networks weights and biases in order for the network to have good perform classifying the inputs, It also still uses the same stochastic gradient descent and backpropagation algorithms.

Illustrated in Figure 2.3) is a convolutional neural network used to classify numbers from 28x28 images. It has 28 x 28 input neurons used to encode the pixel intensities, then is followed by a convolutional layer using a 5 x 5 local receptive field and 3 feature maps which results in 3 x 24 x 24 feature neurons. Next, a max-pooling layer is applied with 2 x 2 regions of stride 2 which results in 3 x 12 x 12 hidden feature neurons. Finally, the output layer has a fully connected structure (which is not fully connected in the figure for simplicity) with 10 neurons, because the objective of the network is to classify digits between 0 and 9 , meaning that it has 10 different classes to classify and therefore should have 10 different output neurons.

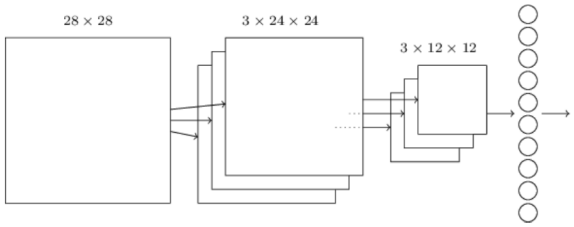


**Figure 2.3:** Convolutional neural network architecture used to classify digits from the MNIST dataset [6]

This represents a simple CNN architecture for a simple problem. However, even in this scenario we can already see that the number of learnable parameters escalates quickly in this type of networks. As such, it is crucial to develop highly efficient and effective architectures for more complex problems.

### 2.1.3 Convolutional neural network architectures

Over the years several CNN architectures have been developed and tested against state of the art benchmark challenges such as the ImageNet Large Scale Visual Recognition Competition (ILSVRC) [7]. In 2012, Krizhevsky et al. [8] submited for the first time a CNN architecture (AlexNet) which outperformed hand-crafted feature learning on the ImageNet. It contained 8 neural network layers, 5 convolutional and 3 fully-connected. This laid the foundation for traditional CNNs: a convolutional layer followed by an activation function followed by a max pooling operation.

Following AlexNet main ideas, the VGGNet[5] was created and became quite popular by winning the 2014's ILSVR. This architecture proved that representation depth is beneficial for the classification accuracy, by using the traditional convolutional network architecture but with increased depth along with smaller receptive fields. There are some public variations of this network, one of which having 16 weight layers (VGG16) displayed in Fig. 1. This architecture is composed by multiple sets of convolutional layers followed by pooling layers that build progressively more abstract features, and at the end a fully connected structure to convert the results of the convolution into a label.

Until this point, both AlexNet and VGGNet use plain networks, which are networks that stack convolutional layers followed by fully connected layers and both approaches explore

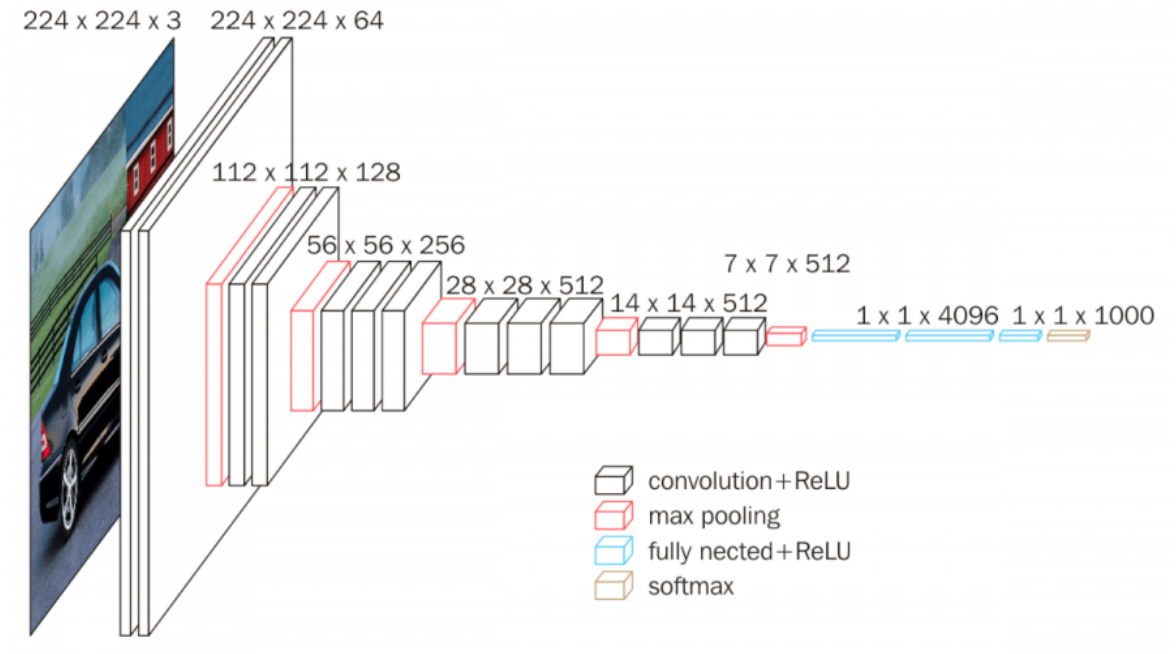


**Figure 2.4:** Architecture of the VGG16 convolutional neural network [5]

the idea of creating deeper networks in order to obtain better performance. However, at a certain point when the network becomes too deep, the problem of vanishing gradients starts to be prominent. The reason behind this problem is that on deep networks as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small. Consequently, as the network becomes deeper, its performance gets saturated or even starts degrading rapidly.

In 2015 a team of Microsoft researchers won the ILSVR, and later submitted a paper to the 2016's CVPR, in which they show that a 20 layer plain network performs better than a 56-layer plain network one due to the presence of the vanishing gradients problem on the deeper network. As such, they attempt to solve this problem with the introduction of a concept called skip connections seen in Fig. x. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlaying mapping.

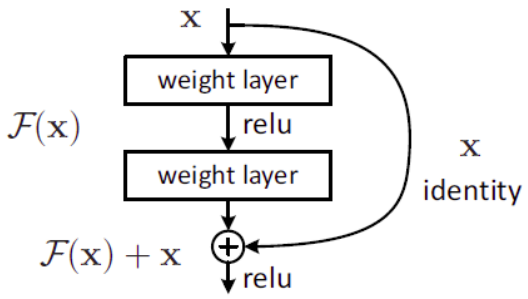


**Figure 2.5:** Building block of residual networks [9]

The fundamental idea behind skip connections is to learn a kind of residual mapping:

$$F(x) = H(x) - x \tag{2.2}$$

The advantage of ResNets is that the gradient can flow directly through the identity function from later layers to earlier layers, which helps with the vanishing gradient problem.

Several ResNets are available with different depths. Some of the most popular variations have 50, 101 and 152 layers which had top 1 accuracy scores on the ImageNet dataset of 0.749, 0.764, 0.766, respectively. However, as Huang et al. pointed out, when the output of a layer l is combined with the identity function by summation, the information flow of the network may be compromised, meaning that some information might be lost in this process. Therefore, they extended the skip connection concept further in their work at [10] with the introduction of a new type of neural network called Dense Convolutional Network (DenseNet).

Unlike ResNets, any layer has direct connections to all subsequent layers, and feature maps are combined with concatenation instead of summation. Authors argue that concatenating feature-maps learned by different layers further improves information flow efficiency and increases variation of subsequent layers. Consequently, the $l$th layer receives the feature-maps of all preceding layers, $x_0, ..., x_{l-1}$, as input:

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{2.3}$$

where $[x_0, x_1, ..., x_{l-1}]$ refers to the concatenation of the feature-maps produced in layers $0, ..., l-1$. Figure 2.6 illustrates the DenseNet schematics.

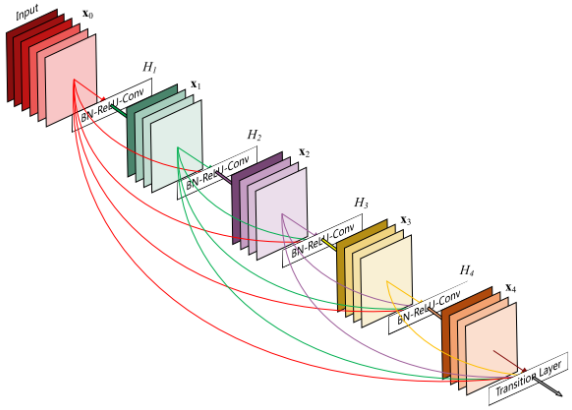


**Figure 2.6:** A block from DenseNet with five layers. Each layer combines all preceding feature maps through concatenation. [10]

Similarly to ResNet, DenseNet provides different versions with different depths, with 121, 169 and 201 layers performing slightly better than ResNet on ImageNet with top-1 accuracies of 0.750, 0.762, 0.773.

So far, the presented architectures explore the idea of scaling up depth of the network in other to build progressively more abstract features, however, there are three dimensions in which one can scale a network (illustrated in Figure 2.7):

- Width scaling controls the width of each layer on the model (Figure 2.7 (b)). Wider networks tend to be able to capture more fine-grained features and are easier to train, but extremely wide, shallow networks tend to have difficulties in capturing higher level features [11].
- Depth scaling controls how many layers the model has. All the former presented models explore the idea of scaling up the network in order to capture richer and more abstract features. However, as we've seen problems emerge from creating deeper networks, specifically, vanishing gradients. Approaches like [9] and [10] attempt to solve this problem but performance gains diminishes as models become deeper. For example, ResNet-101 and ResNet-1000 have very similar accuracy [9].
- Resolution scaling corresponds to the increase of resolution of input images. The intuition behind this type of scaling is that the network can see more detail and therefore capture more fin-grained patterns.

Tan et al. noted these three scaling dimentions are not independent. Rather, intuition tells us that for higher resolution input images, there should be a depth increase such that the larger receptive fields capture similar features that include more pixels in bigger images. It also tells us that we should increase the network width in order to capture more fine-grained pattterns, because the images have more information (more pixels). Following this ideas, they proposed a new family of models called EfficientNets [11] which coordinates and balances different scaling dimentions together through the compound scaling method presented in equation 2.4.

$$\begin{aligned} \text{depth: } & d = \alpha^\theta \\ \text{width: } & w = \beta^\theta \\ \text{resolution: } & r = \gamma^\theta \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2.4}$$

They created a baseline model EfficientNet-B0 and then scaled up the model in three dimentions following those constraints creating 7 other models (EfficientNet-B1 to EfficientNet-B7) which increasingly have more parameters and FLOPS, but also in theory should have increasingly better performance. The idea is to provide a family of models that can be adapted to one's needs, depending on the available compute capability.

Multiple pre trained models trained on ImageNet can be seen in Table 2.1, including the ones referenced so far. https://github.com/albanie/convnet-burden https://keras.io/applications/

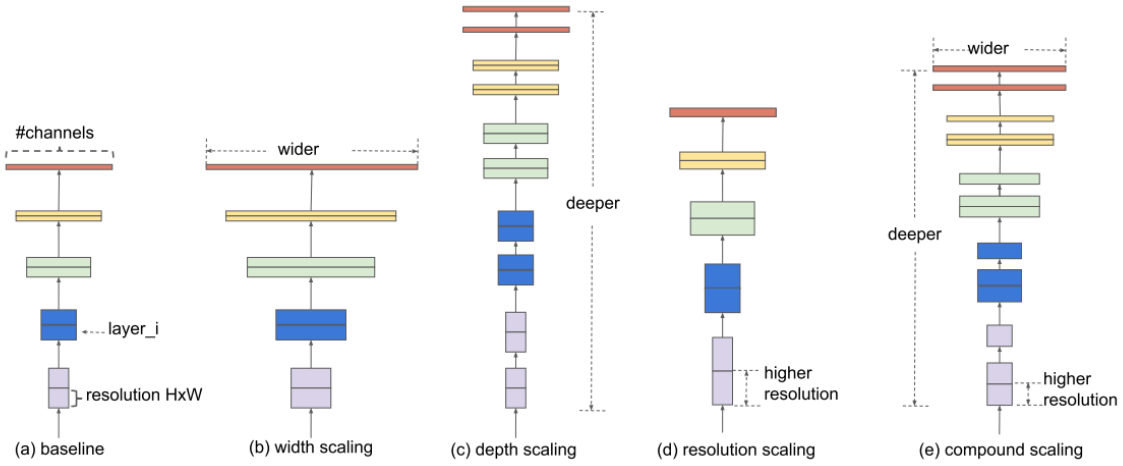


**Figure 2.7:** Model scaling methods. [11]

| Model | Year | Size | Top-1 Accuracy | Top-5 Accuracy | Params | Depth | Input Size |
|---|---|---|---|---|---|---|---|
| AlexNet | 2012 | 233 MB | ? | ? | ≈60M | 8 | 256x256 |
| VGG16 | 2014 | 528 MB | 0.713 | 0.901 | ≈138,4M | 23 | 224x224 |
| VGG19 | 2014 | 549 MB | 0.713 | 0.900 | ≈143,7M | 26 | 224x224 |
| ResNet50 | 2015 | 98 MB | 0.749 | 0.921 | ≈25,6M | - | 224x224 |
| ResNet101 | 2015 | 171 MB | 0.764 | 0.928 | ≈44,7M | - | 224x224 |
| ResNet152 | 2015 | 232 MB | 0.766 | 0.931 | ≈60,2M | - | 224x224 |
| ResNet50V2 | 2016 | 98 MB | 0.760 | 0.930 | ≈25,6M | - | 224x224 |
| ResNet101V2 | 2016 | 171 MB | 0.772 | 0.938 | ≈44,7M | - | 224x224 |
| ResNet152V2 | 2016 | 232 MB | 0.780 | 0.942 | ≈60,4M | - | 224x224 |
| DenseNet121 | 2016 | 33 MB | 0.750 | 0.923 | ≈8,1M | 121 | 224x224 |
| DenseNet169 | 2016 | 57 MB | 0.762 | 0.932 | ≈14,3M | 169 | 224x224 |
| DenseNet201 | 2016 | 80 MB | 0.773 | 0.936 | ≈20,2M | 201 | 224x224 |
| InceptionV3 | 2015 | 92 MB | 0.779 | 0.937 | ≈23,9M | 159 | 299x299 |
| InceptionResNetV2 | 2016 | 215 MB | 0.803 | 0.953 | ≈55,9M | 572 | 299x299 |
| Xception | 2016 | 88 MB | 0.790 | 0.945 | ≈22,9M | 126 | 299x299 |
| EfficientNetB0 | 2019 | 5.3 MB | 0.773 | 0.935 | ≈5,3M | 201 | 224x224 |
| EfficientNetB1 | 2019 | 7.9 MB | 0.792 | 0.945 | ≈7,8M | 201 | 240x240 |
| EfficientNetB2 | 2019 | 9.2 MB | 0.803 | 0.950 | ≈9,2M | 201 | 260x260 |
| EfficientNetB3 | 2019 | 12.3 MB | 0.817 | 0.956 | ≈12M | 201 | 300x300 |
| EfficientNetB4 | 2019 | 19.5 MB | 0.830 | 0.963 | ≈19M | 201 | 380x380 |
| EfficientNetB5 | 2019 | 30.6 MB | 0.837 | 0.967 | ≈30M | 201 | 456x456 |
| EfficientNetB6 | 2019 | 43.3 MB | 0.842 | 0.968 | ≈43M | 201 | 456x456 |
| EfficientNetB7 | 2019 | 66.7 MB | 0.844 | 0.971 | ≈66M | 201 | 224x224 |

**Table 2.1:** Pre-trained models comparison

### 2.1.4 Transfer learning

Training deep neural networks from scratch is often a difficult process because it requires:

- large amounts of labelled data that closely resembles the field it tries to describe
- time to train which is largely dependent on the computational power available
- ability to reason about hyper parameters and follow heuristics in order to achieve a good model on the cross validation process

However, these requirements can be quite difficult to attain specially for small teams with limited monetary and human resources. Particularly in the medical imaging context, public datasets to train deep networks are rather scarce, mainly because of the nature of such data, that expose patients leading to a lack of privacy and anonymity. Even when one has a good dataset along with high computational power, the training process can take a long time, especially while debugging the network to determine a good model fit.

As such, transfer learning emerged as a way to solve this problem, by relaxing the need for the aforementioned requirements. Transfer learning is a method of reusing a pre trained model's knowledge for another related task[12]. In deep learning, using transfer learning means to carry the parameters from pre-trained models such as the architectures presented in Section 3.2 trained on a generic datasets such as ImageNet and using those to train another model with a different purpose.

Transfer learning techniques are usually used to solve image classification problems [13] because it leverages general knowledge from pre trained models towards specific purposed models.

In CNNs, as inputs are passed along the network hidden layers closer to the input layer output generic features like shapes and curves, while hidden layers closer to the output layers build more abstract features such as a dog's face. In order to adapt the pre-trained models into a different domain, one must extract the parameters up to some layer from the pre-trained model while freezing (to not allow parameter updates while training) some or no portion of those layers. As layers near the input layer output generic features, their parameters are usually extracted and potentially frozen, while hidden layers near the output layer are usually not extracted and not frozen, because they output more abstract problem specific features. Figure 2.8 illustrates 3 different strategies to deal When repurposing the knowledge from a pre trained model one must first remove it's classifier, and replace it by a classifier that fits one's needs, while keeping the rest of the architecture (called convolutional base) intact. Figure 2.8 illustrates 3 different strategies to repurpose a pre-trained model:

- Strategy 1: Fine tune the whole model. Often requires both more computational power, due to having more parameters to train, and more data to adapt the model to it's new purpose.
- Strategy 2: Freeze part of the model while fine tuning the remaining layers. As previously mentioned lower layers identify problem independent features, while higher layers refer to problem dependent features. As such, one can chose up to which layer should the model be freezed depending on how much different his problem is compared to the original pre trained model problem. Some general advice can be given about this type of strategy. If

the dataset to train the pre trained model is big enough and enough computational power is available the benefits of fine tuning more layers outweight the cons (namely more time to train the model). However, if one has a big enough dataset and low computational power, only fine tuning higher layers might be preferable. TODOOOOOOO

- Strategy 3: Keep the convolutional base parameters and only train the classifier. Usually used in cases where the purpose of the pre-trained model is very similar to the problem that we are trying to solve, meaning that the datasets are similar. This is can be a especially good strategy if the available computational pwoer is low because the only parameters required to train are the classifier one's.

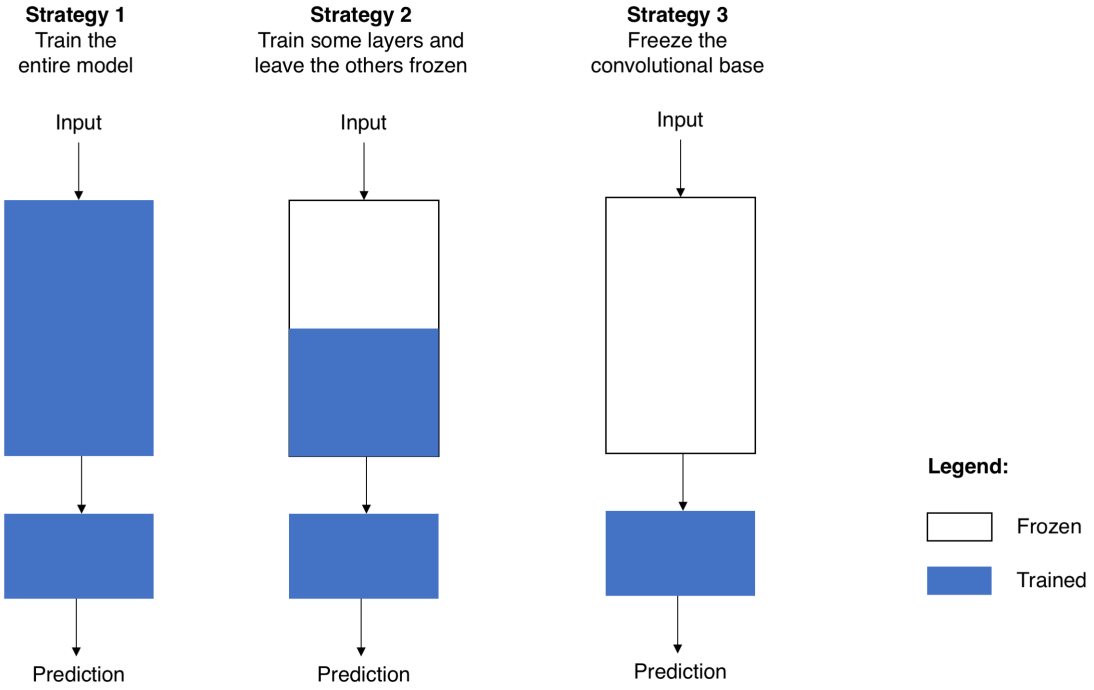


**Figure 2.8:** Transfer learning fine tuning strategies [14]

### 2.1.5 Overfitting and underfitting

While training, one must fine tune the model to both accurately make predictions from the training data while generalizing to new data. The bias and variance trade off is a well known problem in deep learning that represents a trade off between these two requirements. While the bias of a model is the error caused by the assumptions made to approximate the model to the true predictions, the variance of a model is the error from sensitivity to small fluctuations in the training set. We must find a good trade off between bias and variance so that the model doesn't underfit or overfit.

If the model underfits then it does not perform well even on the training data, and therefore has high bias and low variance. However, a common problem is to produce a model that performs well on the training data but that generalizes poorly to new data [15]. In this case, we say that the model overfits and therefore has low bias but very high variance. In order to

evaluate whether a model is underfitting or overfitting one should use state of the art metrics which help describe what is happening while training.

Multiple solutions to the overfitting problem have been proposed and tested over the years. One common way of dealing with this problem are the regularization techniques, which are broadly described by some authors as any technique that allows the model to generalize better. For example, L1 and L2 regularization attempt to create less complex models [16], while techniques such as dropout "reduce complex co-adaptations between neurons" [17]. Other methods such as data augmentation can also minimize this problem.

REFERENCIAR FIGURA!!

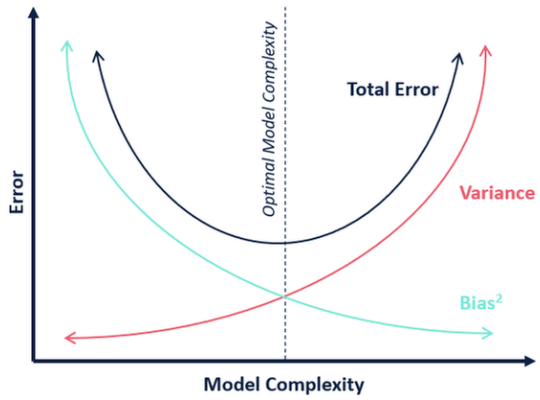


**Figure 2.9:** Influence of bias and variance on total error

*Expanding the training data*

In deep learning, a model is highly dependent on its training dataset in order to achieve good performance. A bad dataset can easily cause the network to overfit because it does not provide enough proper real world examples for the network to produce a good bias variance trade off. A good dataset has to represent the real world it tries to describe, be diverse, and most importantly, it needs to have a good number of examples. There are several datasets available which are labelled for skin lesion diagnosis, but a lot of them are quite biased towards some specific class or lack large amounts of examples for a specific class. As such, when some real world variation is introduced the network fails to predict the class.

One way to improve the training dataset with low costs is through a concept called data augmentation. The main idea behind this concept is to expand the training data by applying operations that reflect real-world variation [6], which in turn introduces diversification and size to the dataset. The simpler approach is to apply general transformations, such as translations, rotations or flips to existing samples to create new ones. Another more complex approach is to synthetically create new images based on some original dataset (generative models) through methods such as generative adversarial networks, a type of neural networks.

### 2.1.6 Ensemble learning

Neural networks are non linear methods, which means that they can learn complex non linear relatioships in the data. A downside of this flexibility is that they are sensitive to initial conditions, both in terms of the initial random weights and in terms of the statistical noise in the training dataset [18]. As such, depending on it's initial conditions it may learn a very different version of the mapping function from inputs to outputs, which will directly impact the model's performance. This means that neural networks tend to have produce low bias and high variance.

One solution to this low bias high variance scenario is to use ensemble learning, which is a machine learning paradigm that combines multiple weak learners (models) that solve a particular problem in order to create a presumably better performing model. These are called weak learners because they often suffer from either high bias which usually happens to low complexity algorithms (e.g. linear regression) or high variance which is associated with high complexity algorithms (e.g. neural networks).

Ensemble learning is a studied field in that provides several methods to combine models. Those combinations are often based on 3 variations.

*Varying the choice of data*

The data used to train each member of the ensemble can be varied. The simplest approach would be to repeatedly sample the dataset with a random splits of the data into train and test sets. We can then train different models based on each of those samples and combine them together.

An alternative would be to perform k-fold cross validation. For this procedure, k different models are trained on k different subsets of the training data. These models can then be saved and used as members of an ensemble.

Another approach is to perform bagging (short for bootstrap aggregating). Given a training dataset with size n, bagging generates m new training sets each of size k, by sampling from the original dataset uniformly with replacement (observations may be repeated). Then, m models are fitted using the m bootstrap samples and are combined by averaging the output for regression problems or voting for classification problems.

*Varying the choice of the models*

Training neural networks with different initial conditions will result in different models. This means that we can vary those initial conditions and combine the resulting models in order to reduce the variance.

A possible approach for this would be to use a different set of hyperparameters (e.g. learning rate or loss function used) or even a different set of model architectures. This will hopefully create an ensemble of models that is going to learn substantially different mapping functions between inputs and outputs, which will lower the correlation between predictions and in turn lower the resulting model variance. An alternative to this method is to periodically save the best model during training, and then ensemble them. This method is

called snapshot ensemble. At the heart of snapshot ensembling is an optimization process which visits several local minima before converging to a final solution [19]. A variation for this method is to save models from a range of epochs, perhaps identified by reviewing learning curves **horizontalvertical**. Stochastic gradient descent with warm restarts [20] is another variation in which the optimization procedure is changed during training (e.g. oscillating the learning rate) and the best models are saved on specific checkpoints.

*Varying the choice of the way that outcomes from ensemble members are combined*

A simple way to combine predictions is to calculate the average between prediction between different ensemble models. A variation of this is to. A slightly more complex approach is to create a new model that learns how to combine the predictions of the different ensemble models, This approach is often called model stacking.

In more sophisticated methods such as boosting, ensemble members are added one at a time in order to correct the mistakes of prior models. Other approaches such as model weight averaging **?** attempt to average the weights of neural networks with the same architecture, rather than their predictions.

### 2.1.7 Deploying deep learning models

Deploying deep learning models requires careful orchestration of components such as a learner for generating models, a visualization tool to analyse and validate models and a serving framework which exposes models through an API. Usually, these components are hard coded together by custom scripts leading to high coupling and low cohesion. This poses problems for future expandability of such systems because simple changes can completely break the pipeline. Therefore, it is a priority to build these systems in a modular approach such that components are independent of each other. In addition, requirements such as easy-to-use configuration and tools, scalability and reliability should also play a big role when considering frameworks and tools to create a cohesive architecture for a production application.

Frameworks such as Tensorflow Extended[21] attempt to integrate the aforementioned components and requirements into one platform and standardize the whole process. Tensorflow has become a more production centered platform over the years, for example, by integrating Keras[22] into it, which provides more easy to use high level concepts to train and test models. There are other options such as Theano[23] or pyTorch[24] but these are more focused around research environments.

Training deep learning models usually requires high computational requirements, which is why nowadays most of these frameworks take advantage of GPUs through the CUDA platform. However, for small teams such computational power might be inaccessible or the cost of either time or money to setup such system might be too much. In such cases, it is better to take advantage of cloud services to train these models.

## 2.2 eHealth/mHealth for skin lesion diagnosis

Online health, ehealth and mhealth applications represent a rapidly developing field of medicine that has the potential to become powerful tool in the diagnosis and management of skin diseases [25]. These applications aim to enhance clinical care, promote health, prevent diseases and most importantly provide medical support when it is not available at a particular location or time. Generally, the acceptance towards this type of systems in the medical community keeps growing, but is highly dependent on factors such as performance, accessibility and ease of use, which poses challenges for their global adoption **?**.

Currently, several production ready skin lesion classification systems are available for both skin professionals and patients wishing to self monitor their own skin. However, almost none of them has shown to be sufficiently accurate or reliable enough for a clinical environment.

This subsection reviews the literature related with the development of dermatology applications for the self-surveillance of skin lesions. These applications offer the patient the ability to monitor their skin condition using images self-captured with a smartphone, while the capability for processing and displaying results is very variable. These efforts have been leveraged by the growing availability of smartphone attachments capable of turning these devices into compact dermatoscopes. Here, a chronological order is followed using a wide range of sources, including the contributions of Brewer et al. (2013), Kassianos et al. 2015 and Zaidan et al. (2018) that organized their surveys as a map of research concepts discussed in relevant literature.

Brewer et al. (2013) identified and categorized the variety of mobile apps available in dermatology, in May 2013, for both the general public and the dermatology providers. A total of 229 dermatology apps were identified in thirteen categories, the most important being the following: (i) general dermatology reference: 61 apps whose contents range from a short list of common skin conditions for the general public to a comprehensive reference guide for medical professionals; (ii) self-surveillance/diagnosis: 41 apps allowing users to document lesions, upload/receive dermatologist or algorithm-based feedback about malignancy potential of lesions, log personal treatment regimens and record symptoms to allergen exposures; (iii) sunscreen/UV recommendation: 19 apps provide sunscreen recommendations based on the skin type and the weather conditions; (iv) teledermatology: 8 apps allow for mobile consultation services; (v) dermoscopy: 2 apps were identified that require a proprietary dermoscope smartphone attachment.

Later, Kassianos et al. (2015) reviewed smartphone applications for the detection of melanoma targeted at the general community, patients and clinicians. The authors reported the existence of 39 dermatology-related applications available on the market, in July 2014, in a range of functionalities including information, education, classification, risk assessment and monitoring change. Over half of them provided information, advice or education about melanoma, ultraviolet radiation exposure prevention advice, and skin self-examination strategies, mainly using the ABCDE (A, Asymmetry; B, Border; C, Colour; D, Diameter; E, Evolving) method. Half of the apps helped users take and store images of their skin lesions

either for review by a dermatologist or for self-monitoring to identify change, an important predictor of melanoma. A similar number of apps provide reminders to help users monitor their skin lesions. A few apps offered expert review of images. Four apps provided a risk assessment to patients about the probability that a lesion was malignant or benign, and one app calculated users' future risk of melanoma. There was little evidence of clinical or research-based input into the design of these apps. Furthermore, none of the apps appeared to have been validated for diagnostic accuracy using established research methods.

More recently, Zaidan et al. (2018) reviewed the literature on smartphone applications for skin cancer diagnosis in the period from 2011 to 2016. A total of 89 articles were classified into four groups: development and design, analytical, evaluative and comparative, and review and survey studies. Out of the 89 articles, 43 focus on the development of various AI algorithms and applications for assisting in the prevention and early detection of malignant melanoma. A total of 20 articles involve analytical studies on the incidence of skin cancer, the classification of malignant cancer or benign cancer and methods for prevention and diagnosis. A total of 15 articles consist of studies that range from evaluation or comparison of mobile apps to the exploration of features designed for skin cancer detection. A total of 11 articles comprise reviews and surveys referring to actual applications or providing a general overview of the technology.

In the meantime, Jaworek-Korjakowska and Kleczek (2018) reported the existence of about 45 mobile applications related to mole diagnosis available on Apple's App Store in March 2017. Most of them offered only educational information on melanoma, nearly half of them allowed the user to take photos of their moles and track changes over time using simple visual comparison, and only four applications performed melanoma risk assessment or lesion classification based on image analysis. These applications are DermaCompare (risk assessment through image matching), Lūbax (mole diagnosis through content-based image retrieval), MySkinApp (risk assessment) and SkinVision (risk assessment through fractal analysis). From these applications, two were certified by authorities, namely SkinVision received the European "CE" Marking and DermaCompare was approved by the U.S. Food and Drug Administration.

One of the most popular eHealth applications for this purpose is Metaoptima's Dermengine web application. Their Visual Search tool compares a user-submitted image with similar images in a database of thousands of pathology-labelled images gathered from other dermatologists. Deep learning techniques are used to search for related images based on visual features such as colour, shape or patterns [26].

Another popular app is the SkinVision which classifies lesions as either low, medium or high risk of skin cancer by using an risk assessment algorithm based on gray-scale images of lesions and their associated fractal maps. It achieves the overall sensitivity of 73%, specificity of 83%, and accuracy of 81%. The positive and negative predictive values were 49% and 83%, respectively [25].

## 2.3 Skin lesion diagnosis

Typically, in order to classify a skin lesion from dermoscopic images one must first segment the lesion, then extract relevant features from the segmented part of the image and finally classify the lesion into a specific class.

Skin lesions are often distinguishable from the surrounding area of the skin due to their color, intensity and texture. Lesion segmentation aims to separate that region from the rest of the image, in order to apply further processing it (e.g., identification of global morphological features specific to the lesion and segmentation of various local features at a later stage). This can become a challenging problem because demoscopic images often come with various artifacts, like hairs, ruler marks, date markers, ink, bubbles etc. and in many cases there is a low-contrast between the bounded region and the surrounding area (due to the lighting conditions, skin tone variations and so on). Additionally, the lesion itself can have variations in color, texture, shape, size and location that make it difficult to contrast it with the background. Although the impact of these factors can be minimized with proper pre-processing.

Therefore, both high and low level methods for performing lesion segmentation have been proposed over the years, including histogram thresholds, clustering, active contours, edge detection, graph theory and probabilistic modeling **?**.

With the bounded region one can start feature extraction by identifying points of interest within that region. Such interest points are often attributes like color, texture, shape, structure, relative size and location of the lesion [27]. As a result of several studies, multiple algorithms have been identified to help determine whether a given skin lesion is melanocytic/nonmelanocytic and benign/malignant, namely:

- Pattern analysis (Pehamberger and Steiner, 2016; Barzegari et al., 2005) identifies melanocytic lesions using local and global patterns. Local patterns include pigment system, dabs and globules, streaks, blue whitish shroud, relapse structures, hypopigmentation and vascular structures. Global patterns include reticular, globular, cobblestone, homogeneous, stardust, parallel, multi- component, lacunar and unspecific patterns.
- ABCD-rule (Nachbar et al., 1994) evaluates melanoma diagnosis using four criteria which include asymmetry, border irregularity, color and diameter. Each criteria is multiplied by a weight factor to establish a Total Dermoscopic Score (TDS) such as a lesion is considered benign for a TDS score lesser than 4.75, an intermediate TDS value of 4.75–5.45 it is regarded as possible melanoma and a score above 5.45 indicates probable melanoma.
- Menzies method (Robert, 2002) is based on a set of negative and positive features. Negative features include patterns that are point or axial asymmetric and presence of a solitary shading (gray, ebony, blue red, tan and dark brown). The nine positive features include blue white veil, multiple brown dots, pseudopods, radial streaming, depigmentation, peripheral ebony dots/globules, multiple hues, numerous blue/dark spots and expanded network. For a lesion to be classified as melanoma at least one positive feature must be found.

- 7-point checklist (Unlu, Akay and Erdem, 2014; Walter et al., 2013) assigns a score based on three major (atypical pigment network, blue-white veil and atypical vascular pattern) and four minor criteria (irregular streaks, irregular pigmentation, irregular dots/globules and regression structures). To score a lesion, the proximity of a major criterion is given 2 points and that of a minor model is given 1 point. The lesion is named as melanoma when the aggregate score is more prominent than or equivalent to 3.
- CASH algorithm (Henning et al., 2007) is designed as a simplified form of pattern analysis in which lesions are evaluated according to the color, architecture, symmetry and homogeneity.

Finally, the image features extracted can be fed to a classifier that should be able to distinguish, for instance, melanomas from benign lesions. Such classifier can be based on a multitude of algorithms such as artifitial neural networks or support vector machines. Figure 2.10 illustrates the distribution of the most used algorithms for classifying features from skin lesions.

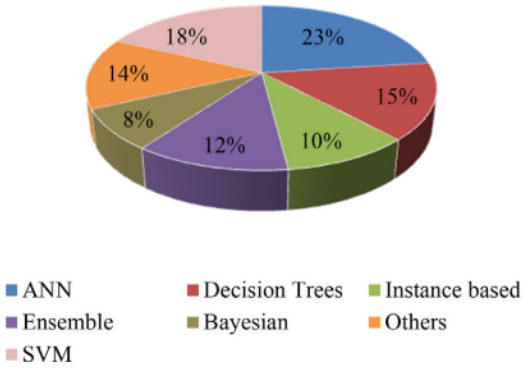


**Figure 2.10:** Illustration of classification methods used by existing diagnostic methods [28]

While these type of approaches provide a way of classifying skin lesions, they require domain knowledge, particularly, when performing segmentation and feature extraction. More recently end-to-end learning techniques have become prominent in fields such as medical imaging (and for skin lesion diagnosis in particular), meaning that no preliminary steps such as lesion segmentation or feature extraction are necessary. These techniques are often based on deep neural networks **?**.

## 2.4 Deep neural networks for medical imaging

Deep learning refers to computational models composed of multiple processing layers capable of learning representations of data with multiple levels of abstraction [29]. The initial impact of deep learning for medical imaging was revealed through a special issue published in 2016 at the IEEE Transactions on Medical Imaging [30]. It explains the principles and methods of deep learning applied to medical image analysis. These structures can be found in

approaches to medical imaging problems such as organ segmentation, lesion detection and tumor classification.

The main advantage of deep learning over other machine learning algorithms is that it removes the need for feature engineering, a process that requires knowledge of the problem domain, which can be a time consuming process as well as introduce human error.

Recently, deep neural networks appear as state-of-the-art solutions for medical imaging problems due to advancements in the field. These advancements include the research and development of new methods to prevent overfitting, the rise of computational power along with the use of graphical processing units, and finally the development of high level modules such as theano [23] that help train and test neural networks .

## 2.5 Skin lesion classification using deep learning

One of the most important factors which determines the performance of a deep learning model is the dataset used to train it on **?**. For general image recognition problems datasets such as ImageNet [31] which contains over 14 Million samples with over 20000 classes are usually used and serve as a benchmark. However for skin lesion diagnosis systems, it is difficult and in many times impossible to compare the performance of published classification results since many authors use nonpublic datasets for training and testing [32]. The closest benchmark dataset available for this domain is the HAM10000 dataset [33] and consists of 10015 dermatoscopic images and contains classes such as Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc).

Nonetheless, organizations such as International Skin Imaging Collaboration (ISIC) provide an open source public access archive of skin images, which can be used for teaching or for the development and testing of automated skin lesion diagnosis systems [34]. They also place a challenge around their dataset every year in order to improve the performance of this classification systems as a whole.

### 2.5.1 Transfer learning approaches

Perhaps one of most popular approaches for skin cancer classification using deep learning, specifically, convolutional neural networks, was published in 2017 by Esteva et al.[3], in which their classifier could diagnose keratinocyte and melanoma cancer. Their dataset combines biopsy-proven data from the ISIC archive, Edinburgh Dermofit Library, and the Stanford Hospital, totalling an astonishing 129450 samples (after going through data augmentation of random flips, rotations, crops) which remains one of the biggest efforts in data collection in the area. The authors follow a transfer learning approach by leveraging the weights of the InceptionV3 network trained on ImageNet, on top of which they build their own classifier. Finally, they measured the network's performance by pitting it against 21 dermatologists on

biopsy-proven samples and concluded that their classifier had comparable performance to that of those board-certified dermatologists.

Haenssle et al [4] presented a very similar approach to Esteva et al. They also fine tuned the Inceptionv3 model, but the analysis was limited to samples of melanoma versus benign nevi. However, they compared their approach with 58 dermatologists which is the largest number of dermatologists involved in a publication about skin lesion classification [32]. They achieved AUC ROC of 0.86, a slightly lower score than that of Esteva et al which stayed at O.94.

More recently,

Providing information and treatment options for a lesion, and detecting skin cancer with a reasonable sensitivity and specificity are the two main goals behind skin lesion classification on a clinical context. While the first goal is a multi-class problem (diagnosis out of range of classes), the second is a binary one ("biopsy" or "don't biopsy"). Contrarily to previous years where the presented challenges focused on binary problems (second goal), in the part 3 of the ISIC 2018 challenge participants were asked to develop a classifier to distinguish between 7 different types of skin cancer which focuses on the first goal. Those classes are:

- Melanoma
- Melanocytic nevus
- Basal cell carcinoma
- Actinic keratosis
- Benign keratosis
- Dermatofibroma
- Vascular lesion

Participants were ranked based on the normalized multiclass accuracy (shown in equation 2.5 as it is closer to real evaluation of a dermatologist [35], but other metrics such as accuracy, F1 score or AUC are computed for scientific completeness. An advantage of this metric is that algorithms can then be compared with physicians performance in skin lesion classification.

$$B = \sum_{i}^{n} \frac{P_i}{n} \tag{2.5}$$

Where $P$ is the precision (or recall), which is given by equation 2.6:

$$P = \frac{TP}{TP + FP} \tag{2.6}$$

The top 3 submissions had balanced accuracies of about 88,5%, 88,2%, 87,1% respectively and were all submitted by Aleksey Nozdryn-Plotnicki and Yolland which work at Metaoptima (the company behind Dermengine) [36]. To train those models they used the provided dataset, along with samples from ISIC archive and other proprietary data, that then they preprocessed by normalizing the samples with the Shades of Gray method [37]. Additionally, they augmented the training data by performing random horizontal flips, random rotations, changes in brightness, saturation, and contrast. They used transfer learning from several pre

trained models trained on ImageNet (such as InceptionV3 or ResNet) and then ensembled the best performing ones [36].

Noteworthy are also the submissions by:

- Gessert et at. [38] has a particularly interesting approach for inference (shown in Figure 2.11), while performing effectively on task 3 with a balanced multiclass accuracy of 0.83 (fourth place). SENet154, ResNeXt101, Densenet201, Densenet161, Densenet169, SE-Resnet101 and PolyNet are the pre-trained models used for ensembling. For each architecture 6 models are trained, 5 by performing 5-fold cross validation and 1 by training it with the whole dataset (without validation set). Evaluation is performed by cropping 36 patches from each test sample and feeding those patches to each model. Then, for the models that do not have a validation set, the softmax probabilities are averaged across all 36 patches. and for models that were cross validated, predictions are combined using an Support Vector Machine. Finally, both results are combined by averaging the softmax probabilities.

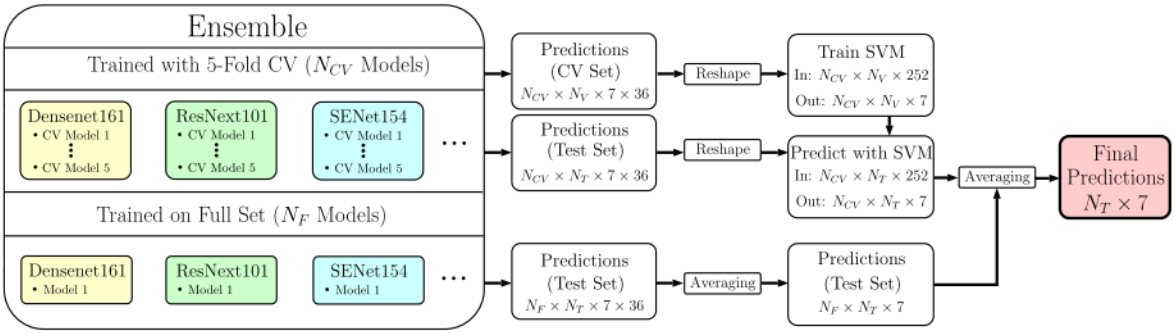


**Figure 2.11:** Gessert et al's evaluation strategy for the generation of the final predictions. [38]

- Ashraful Alam Milton [39] who also followed an approach based on transfer learning from models of PNASNet-5-Large, InceptionResNetV2, SENet154, InceptionV4 trained on ImageNet, who interestingly noted that in the first few epochs the gradient is very erratic and thus refrained from fine-tuning weights during the first 2 epochs to avoid updating weights towards the wrong direction, in the end achieving a score of 0.76 on the validation set;
- Bissoto, Perez, Ribeiro, et al. [40] (who won 3rd place in the 2017 edition) which transfered knowledge from models of InceptionV4, ResNet-152, and DenseNet-161 trained on ImageNet, by training with online data augmentation (e.g., random crops, flips, rotations, shears, color transformations), SGD with the learning rate being decreased by a factor of 10 whenever validation loss didn't improve for 10 epochs, eventually building an average of 15 models trained only with the challenge data that attained a score of 0.803.

Tschandl et al [41] compared the diagnostic accuracy of 139 machine-learning algorithms from ISIC 2018 with 511 human readers, with the large majority being board-certified dermatologists (55.4%), 23.1% being dermatology residents and 16.2% being general practitioners.

Machine learning algorithms outperformed human readers on the large majority of measures, for instance, in sets of 30 randomly selected lesions, the best machine-learning algorithms achieved a mean of 7.94 more correct diagnoses than the average human reader, and a mean of 6.65 more correct diagnoses than expert readers. However, authors advocated that higher accuracy does not necessarily mean better clinical performance or patient management For instance, these algorithms were trained to optimise the mean sensitivity across all classes, and did not consider that it is more detrimental to mistake a malignant for a benign lesion than vice versa. They also found that the difference between human experts and the top three algorithms was significantly lower for images in the test set that were collected from sources not included in the training set (out of distribution samples), and noted this might be a possible limitation of these algorithms which should be addressed in future research. Other authors such as Han et al, also noted this same limitation in their study [42].

As a response to this unresearched matter, ISIC 2019's challenge asked participants to classify dermoscopic images among nine different diagnostic categories, with 8 known classes (i.e. Melanoma, Melanocytic nevus, Basal cell carcinoma, Actinic keratosis, Benign keratosis, Dermatofibroma, Vascular lesion, Squamous cell carcinoma) and one "unknown" class which conceptually means none of the other classes. Similarly to the 2018's version participants could use their own data to improve the network's performance and were ranked based on a balanced multiclass accuracy [34].

The best submission was done by Geesert et al. (which was the team that achieved fourth place at ISIC 2018 Task 3) scoring 0.636 balanced multiclass accuracy [43]. In adition to the challenge's dataset, 995 dermoscopic images from the 7-point dataset **?** and 1339 images from an in-house dataset were used for training. More specifically, the in-house dataset also contains additional images that were used for the unknown class Before training images need to be preprocessed which is performed by cropping the images, performing image binarization, then applying the shades of gray color constancy method [37] and finally resizing the images (longer side to 600 pixels while keeping the aspect ratio). Data augmentation is also applied before training by randomly changing brightness, contrast, rotation, scale, shear, flip and finally by adding Cutout [44] holes. The authors opted to use two different input strategies, which leads to having different models each with different configurations. The first strategy takes a random crop from the preprocessed image, but the second strategy randomly resizes and scales the image when taking a crop. Like the previous attempts they use a transfer learning approach relying on ImageNet pre-trained models, namely, multiple versions of EfficientNets and a SENet154 which was allegedly added for architecture variability. All models were trained for 100 epochs using the Adam optimizer [45] and a weighted cross-entropy loss function to deal with class unbalance. Predictions for each model are made based on which input strategy was used. The final prediction is made using an ensemble of a subset which contained all the best performing models.

At second place, Zhou et al's [46] achieved a balanced multiclass accuracy of 0.607. Unlike, Gessert et al. no additional data was used for training and testing. Images are resized but maintain the aspect ratio (with the shorter edge having 600 pixels) and the shades of gray

color constancy method [37] is performed. Images are augmented using random cropping, scaling, rotations and color transformations. Like most submissions, they used a transfer learning approach by fine tuning Densenet121, se-resnext50, se-resnext101, EfficientnetB2, EfficientnetB3 and EfficientnetB4 which were all pre trained on ImageNet. 5-fold cross validation was performed with a 80-20 split, and models were trained for 90 epochs with the Adam optimizer [45] with learning rate 5e-5 but decreased by a factor of 2 after 10 epochs. To address the unbalanced class distribution, the authors trained all networks with a weighted loss function where weights are determined using the inverse frequency of classes in the training data. For the ensemble, multiple subsets of models were tested in order to optimize the normalized multiclass accuracy values, but for the final submission EfficienetB3, EfficienetB4 and Seresnext101 were used. During inference, for each test image, N (either 16 or 25) ordered same-sized patches are fed into each network and the final prediction is obtained by averaging the prediction probabilities. Finally, the unknown samples are handled by simply classifying the images whose top-1 probability are less than 0.35 as unknown.

At third place Pollastri et al. [47] uses a more complex approach for dealing with the unknown class. Two models were created in their approach called baseline model A and B with the following procedure:

- Preprocessing is done by padding samples with reflection in order to rescale them them to a square size of 512x512 pixels. Pixel intensities are normalized to have 0 mean and 1 standard deviation.
- Training is performed using a cross entropy loss function which is weighted according to the inverse prior probability of each class, which according to the authors helps with the data unbalance problem in the sense that avoids situations where the network performs a maximum likelihood optimization, where it becomes biased towards the most probable class.
- For baseline model A, DenseNet201, ResNet152 and SeResNext101 were the pre trained models used, while for baseline model B, SeResNext-50, SeResNext-101 and the SeNet-154 were the preferred models.
- Both baseline models use online data augmentation. Baseline model A uses operations such as rotations, flips, Gaussian blur, adding Poisson noise, gamma contrast changes, Cutout augmentations [44] (0 to 3 holes with different sizes), hue/saturation changes. These are then combined in different ways to form an ensemble composed of multiple pre-trained models with different architectures and different data augmentation methods. In contrast, baseline model B uses similar operations but the augmentation configuration is the same across all models.
- Inference in baseline model A is performed by ensembling the results from 21 models (each trained with different pre-trained models and augmentations) for a given sample, but test samples also go through the same augmentations as the ones performed during training. For baseline model B a snapshot ensemble technique [19] was used.
- Their proposition to determine the unknown class is to use the Out of Distribution Detector by [48]. 8 CNNs (Densenet-201) are trained to classify if a given sample is part

of the training dataset or if it is an out of distribution sample. Each model employs 7 classes as inside distribution data, and 1 class as out of distribution, but the left out class is alternated in each of them. Inference is performed in a voting scheme where each network votes whether a given sample is part of training set or not.

Baseline model A and B are then combined together for the final submission which scored a balanced multiclass accuracy of 0.593.

Hsin-Wei Wang approach [49] is particularly note worthy because the materials for replicating his submission are available on Github. His method achieved a balanced multi-class accuracy of 0.505, a substantially worse performance than the best approaches mainly due to the poor results of the out of distribution detector on the unknown class. These are the main aspects of his methodology:

- In addition to the provided challenge dataset, 134 samples were obtained from the ISIC archive and the 7-point dataset to be used as unknown samples during testing.
- Images were resized to fit the network's input size and normalized by per channel mean and standard deviation, which were calculated over the entire training set.
- Similarly to the best approaches an ensemble of models was used which was created by taking the average of softmax probabilities made by three models (pre trained with the ImageNet dataset), the DenseNet201, Xception and ResNeXt50 . During training, first all layers are frozen first for weight initialization of the classifier for 3 epochs, then all the layers are unfrozen and fine tuned during 100 epochs with the Adam optimizer [45] and with a learning rate of 10e-5 which is reduced by a factor of 10 if the validation loss has stopped improving for 8 epochs.
- In order to prevent overfitting, online data augmentation is performed with random crops, random flips, random rotations, random changes on brightness, and finally random changes on saturation.
- Finally, for the unknown class the author took advantage of an already implemented method called ODIN (Out-of-Distribution detector for Neural networks) [50] which depending on 3 parameters (temperature scaling, perturbation magnitude and threshold) and based on the softmax predictions, could determine if a given sample is part of the training classes or not (in/out of distribution),

### 2.5.2 End to end learning approaches

The most common approach to skin lesion classification is through transfer learning. However, end to end learning can make sense in specific contexts. For instance, when data and computational resources is not scarce.

In 2019, Ly et al. [13], trained multiple models from scratch with the intention of deploying such models for offline usage in smartphones. They justified this decision by arguing that using pre trained models with large neural network architectures requires a lot more parameters than models trained from scratch. Their best model attained 86% accuracy, significantly better than other transfer learning approaches, while being much more compact (29M). However, they used a huge dataset titled "PHDB" which was composed of multiple other datasets and contained 80,192 labeled images, which explains the high performance.

CHAPTER 3

# Environment

This section will describe the environment used for both the research and deployment of the deep learning models, namely:

- The dataset used to train the network.
- The preprocessing steps to allow loading the images into the networks.
- The data augmentation techniques used to increase the dataset.
- The software stack used for training/testing models, as well as the web stack for the deployment of such models.
- The hardware used for this research.

## 3.1 Data

The International Skin Imaging Collaboration (ISIC) 2019 challenge [34] provides a training dataset with 25331 samples distributed across 8 different categories. Such categories are: melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, dermatofibroma, vascular lesion, squamous cell carcinoma and finally benign keratosis (a category which includes samples of solar lentigo, seborrheic keratosis and lichen planus-like keratosis). This dataset is composed by samples of HAM10000, BCN 20000 and MSK, all of which are labelled datasets of skin lesions.

The Human Against Machine with 10000 images (HAM10000) [33] dataset is an effort to boost research on automated diagnosis of dermatoscopic images that focuses on the quality and reliability of a large volume of data. All of the samples are manually reviewed by dermatology professionals and non-dermatoscopic images or unreliable diagnoses are filtered out. Finally, all of the samples have the same resolution of 600x450 pixels centered around the lesion. The quality and volume of the data provided by this dataset is remarkably good and allows deep learning researchers to focus on developing reliable models rather than focus on extensive pre-procecssing methods before training.

The HAM10000 had already been used in the task 3 of the ISIC 2018 challenge. The literature suggests that the top scoring approaches of the ISIC 2018 surpassed expert dermatologists **?**, however, as Tschandl et al. pointed out, the algorithms performed worse on images from other dermascopic data sources which were not present on the HAM10000 dataset. This statement shows that the HAM10000 dataset alone does not provide enough samples that are representative of real world scenarios, which might be composed of lesion with different pingmentations and the images might be taken under different conditions (e.g. angle, luminosity or contrast variations).

As a contermeasure, the ISIC committee added two more datasets to the ISIC 2019 challenge, which would hopefully create enough variation and help ISIC challenge metrics be representative of real world scenarios. One of the new datasets is the BCN20000 [51], which contains hard to diagnose images of size 1024x1014 captured between 2010 and 2016 by the Hospital Clinic in Barcelona. These images are often not correctly segmented, located in hard to diagnose locations such as nails or mucosa, and can even be hypopigmented, meaning that the overall color of the lesion is lighter than the skin tone. The resulting database includes 19424 manually revised dermoscopic images corresponding to 5583 skin lesions.

Finally, the other new dataset is the MSK [52] dataset, which contains images of multiple resolutions and multiple aspect ratios. The MSK is composed by multiple datasets, namely:

- MSK-1 which contains Both benign and malignant melanocytic lesions. Almost all diagnoses were confirmed by histopathology reports; the remainder consists of benign lesions confirmed by clinical follow-up. Images were not taken with modern digital cameras.
- MSK-2 composed of Biopsy-confirmed melanocytic and non-melanocytic skin lesions. This dataset includes over 500 melanomas. Many images have polarized and contact variants.
- MSK-3 containing Assorted images and lesions, mostly nevi and basal cell carcinomas. These images were found based on a search not filtered for any particular pathology. All diagnoses confirmed by histopathology.
- MSK-4 having images found based on a search for patients with a personal history, clinical diagnosis, or differential diagnosis of melanoma. All diagnoses confirmed by histopathology.
- MSK-5 composed of seborrheic keratoses obtained from patients during a clinical visit. These lesions were not biopsied and were determined to be seborrheic keratoses by agreement of three experts.

By looking ay random samples of the eight classes of ISIC 2019 dataset from Figure 3.1, one can see how different samples are from one another. Different resolutions, aspect ratios, luminosity conditions are present and in some cases, lesions are segmented around a black background. Even within a particular category, a uninstructed person has a hard time to point out common features between the samples. As a result, this shows how the ISIC 2019 dataset represents a more real world scenario where image samples from skin lesions are not perfectly segmented and pre-processed. Overall, this dataset better approximates a

dermatologists work, which often needs to deal with very heterogeneous samples that were taken under different conditions.

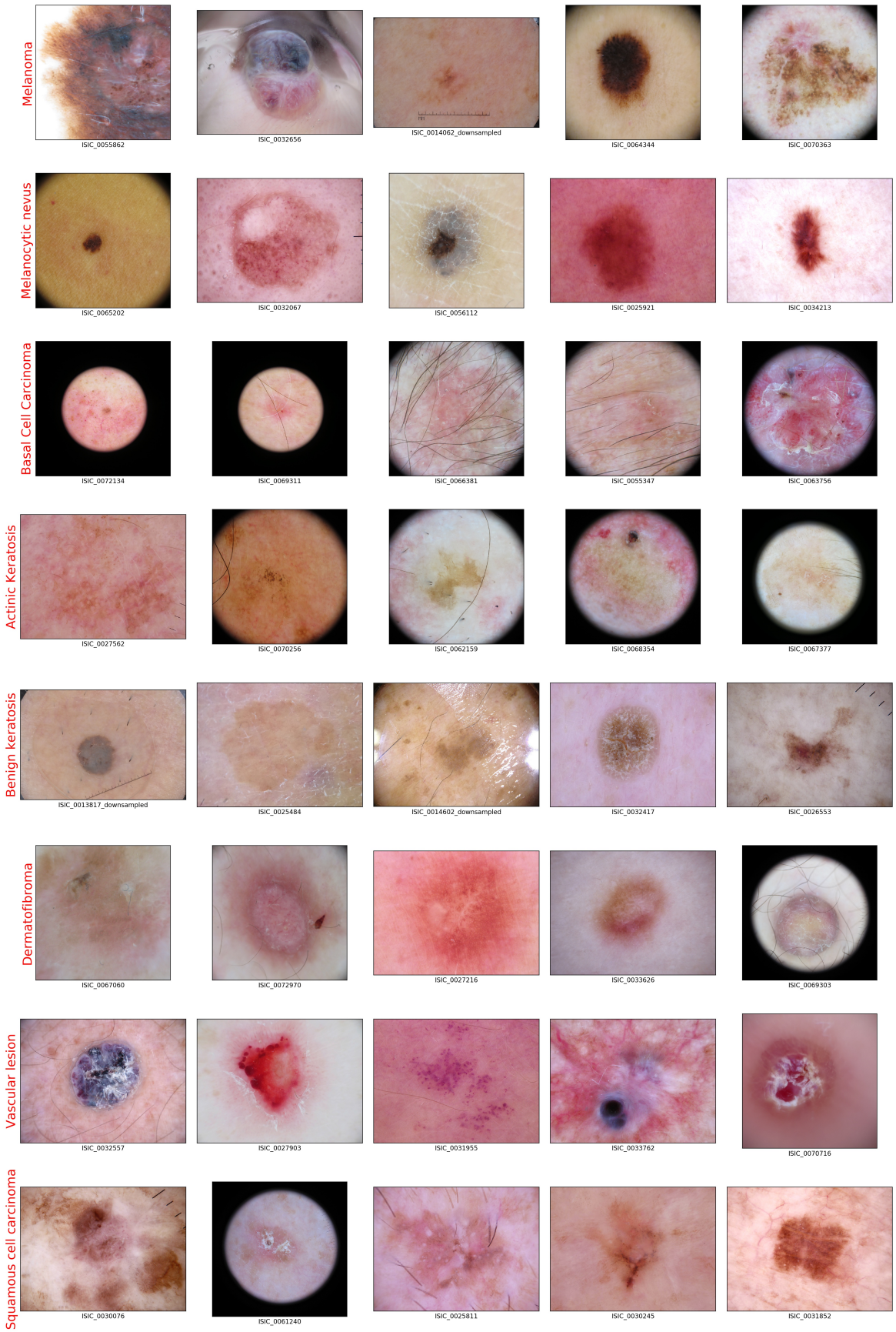


**Figure 3.1:** Samples from ISIC 2019 training data for the 8 known categories

### 3.1.1 Class imbalance

Note that the dataset provided is highly unbalanced with some classes like the melanocytic nevus representing almost half of the whole dataset, while a class like Dermafibroma representing less than 1% of the dataset On chapter ???? we will present study the effects of this class unbalance and present some strategies to deal with it.

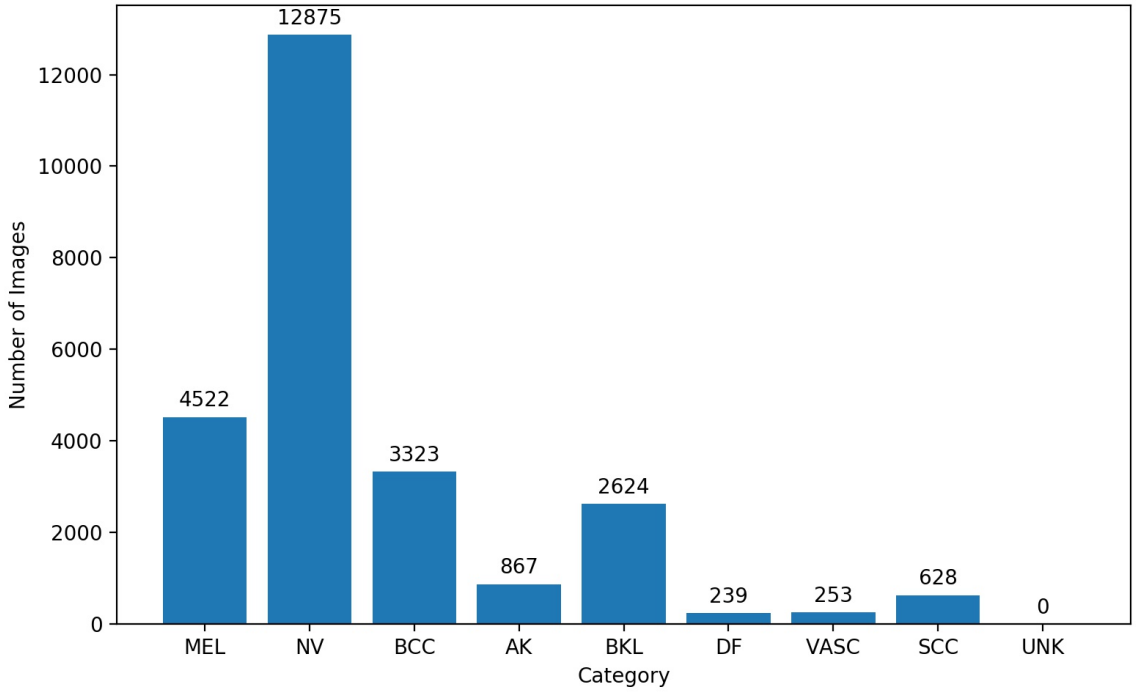


**Figure 3.2:** Class distribution of ISIC 2019 training dataset

To tackle this issue, a common method from the literature **?** is to use some kind of weighted loss function, like the weighted cross-entropy loss function. The weight $W_i$ of each class $i$ is defined by the equation 3.1.

$$W_i = \frac{N}{C * n_i} \tag{3.1}$$

where $N$ denotes the total number of samples in the training set, $n_i$ is the number of samples for category $i$, and $C$ is the number of categories.

Figure 3.3 illustrates that underrepresented classes like dermatofibroma have a considerably higher weight than classes with more samples like melanocytic nevus. This means that the added loss from a misclassified sample of dermatofibroma will have a much bigger impact than a misclassified sample of nevi. Therefore, this method will avoid situations where the model tries to optimize weights towards overrepresented classes like nevi, rather than optimizing performance for each class individually.

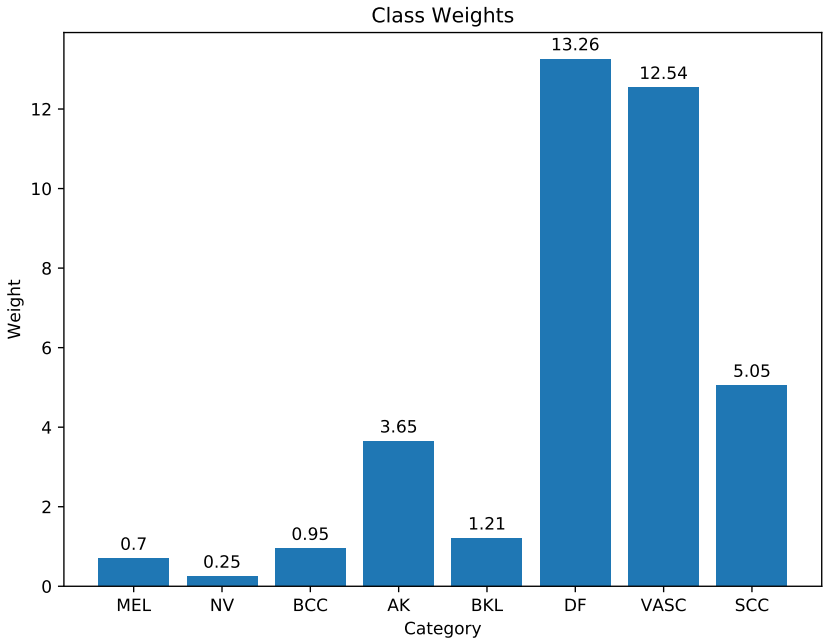


**Figure 3.3:** Weights for each class of the weighted cross-entropy loss function for the ISIC 2019 training dataset

### 3.1.2 Unknown Class

By looking at the the sample distribution across the different classes on Figure 3.2 one can observe that no data is provided for the 9th category, namely, the "Unknown" category. This category is meant to represent none of the other classes, and the idea behind the introduction of this class is to model a more real world scenario.

The range of skin lesion diagnosis extends beyond the 8 classes represented in the ISIC 2019 dataset (see Figure 3.2). As such, when patients show a specific lesion to their dermatologist to diagnose, the lesion interpretation has to consider which type of lesion it represents, possibly not being part of the 8 categories represented in this dataset or possibly not representing a lesion at all, For example, it might be a scar, a bruise or even normal skin. No data is provided for this class because it does not represent a specific category but rather an agglomerate of any skin fraction that is not any of the other 8 lesion categories.

Different strategies to deal with this unknown class will be experimented with in chapter 4. However, for the analysis of the effectiveness of the experimented methods, an agglomerate of multiple samples was created to be categorized as unknown samples. Table 3.1 shows the distribution of such samples.

Most of the samples come from the ISIC archive and are left out from the ISIC challenge because they do not belong to any of the 8 classes known classes. More specifically, they are part of the MSK dataset, which means that samples follow the same conditions to the other samples from same dataset, which often makes them quite similar to images from the ISIC 2019 challenge training dataset for an uneducated observer (see Figure 3.4). As such, these samples will receive the same pre-processing as the samples from the training dataset provided by ISIC.

| Lesion category | Samples amount | Source dataset |
|---|---|---|
| Angiofibroma or fibrous papule | 1 | ISIC Archive |
| Scar | 1 | |
| Angioma | 12 | |
| Atypical melanocytic proliferation | 12 | |
| Lentigo simplex | 22 | |
| Lentigo NOS | 70 | |

**Table 3.1:** Samples per category of out of distribution samples for the unknown class

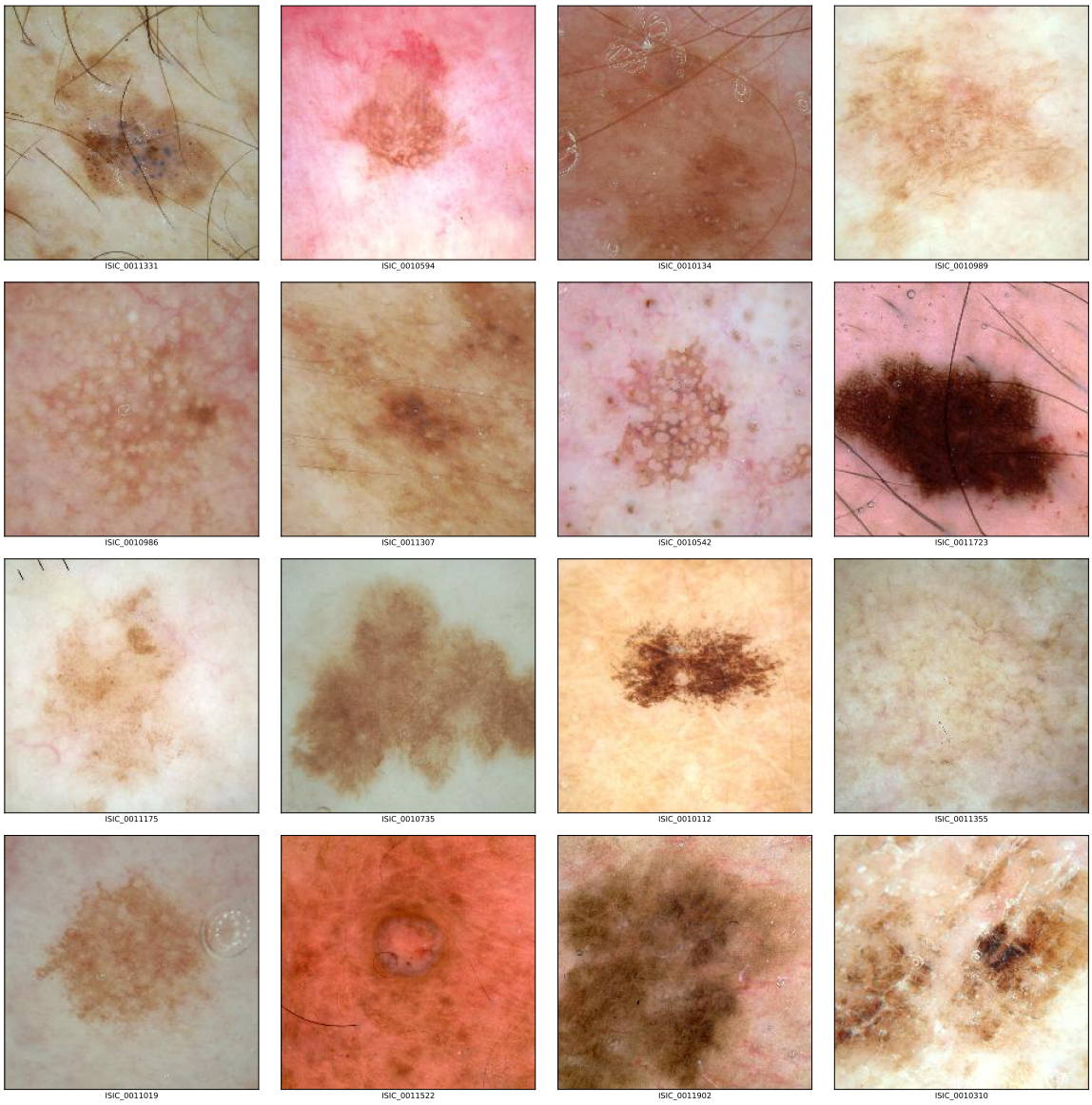


**Figure 3.4:** Randomly chosen samples from the dataset created for the unknown class

### 3.1.3 Preprocessing

Each sample of the the dataset undergoes a number of preprocessing steps:

1. Most readily available pretrained models are of network architectures whose input

tensor is of square dimensions (e.g., $224 \times 224 \times 3$). Since the dataset's images are of distinct non-square dimensions, it is necessary to resize them to a square. However, resizing them all naively to the network's input tensor dimensions without regards to the image's aspect ratio means that the input fed to the network is of varying distinct aspect ratios which does not constitute a good start. Therefore, the first step is to crop an arbitrarily-sized square of the center of the image (as per code snippet 1) which will likely capture the skin lesion, as most images in the training dataset are centered around the lesion.

```
def _crop(img):
    width, height = img.size
    if width == height:
        return img

    length = min(width, height)

    left = (width - length) // 2
    upper = (height - length) // 2
    right = left + length
    lower = upper + length

    box = (left, upper, right, lower)
    return img.crop(box)
```

**Código 1:** Function that crops a given image to a square crop of the center of the original image.

2. Most input tensors from pre-trained model architectures are $224 \times 224 \times 3$, however, more recent architectures such as EfficientNetB2 or InceptionV3 use bigger images to capture more detail. Therefore, images should be resized to the target networks' input dimensions, as soon as possible in the data pipeline in order to reduce the computational costs of any subsequent operations on the images. Images are resized using nearest-neighbor interpolation, as per code snippet 2.

```
def _resize(img, target_size):
    return img.resize(target_size, PIL.Image.NEAREST)
```

**Código 2:** Function that resizes a given image to the target dimensions.

3. Based on Gessert et al. of ISIC 2018 task 3 challenge approach [38], each image is normalized by subtracting the channel-wise mean of the entire ISIC 2019 training dataset, and then dividing it by the it's standard deviation (as in code snippet 3). The channel-wie mean and standard deviation of the ISIC 2019 training dataset was calculated beforehand.

```python
def preprocess_input(x, data_format=None):
    """Preprocesses a numpy array encoding a batch of images. Each image is normalized by subtra
    This function only implements the 'torch' mode which scale pixels between 0 and 1 and then w

    # Arguments
        x: a 3D or 4D numpy array consists of RGB values within [0, 255].
        data_format: data format of the image tensor.
    # Returns
        Preprocessed array.
    """
    if not issubclass(x.dtype.type, np.floating):
        x = x.astype(K.floatx(), copy=False)

    # Mean and STD calculated over the Training Set
    # Mean:[0.6236094091893962, 0.5198354883713194, 0.5038435406338101]
    # STD:[0.2421814437693499, 0.22354427793687906, 0.2314805420919389]
    x /= 255.
    mean = [0.6236, 0.5198, 0.5038]
    std = [0.2422, 0.2235, 0.2315]

    np.mean(x, axis=(0, 1))
    np.std(x, axis=(0, 1))

    if data_format is None:
        data_format = K.image_data_format()

    # Zero-center by mean pixel
    if data_format == 'channels_first':
        if x.ndim == 3:
            x[0, :, :] -= mean[0]
            x[1, :, :] -= mean[1]
            x[2, :, :] -= mean[2]
            if std is not None:
                x[0, :, :] /= std[0]
                x[1, :, :] /= std[1]
                x[2, :, :] /= std[2]
        else:
            x[:, 0, :, :] -= mean[0]
            x[:, 1, :, :] -= mean[1]
            x[:, 2, :, :] -= mean[2]
            if std is not None:
                x[:, 0, :, :] /= std[0]
                x[:, 1, :, :] /= std[1]
                x[:, 2, :, :] /= std[2]
    else:
        x[..., 0] -= mean[0]
        x[..., 1] -= mean[1]
        x[..., 2] -= mean[2]
        if std is not None:
            x[..., 0] /= std[0]
            x[..., 1] /= std[1]
            x[..., 2] /= std[2]
    return x
```

**Código 3:** Function that normalizes the sample images over the entire ISIC 2019 dataset

### 3.1.4 Augmentation

Data augmentation is performed using a different range of techniques which it's impact will be explored in chapter 4. As illustrated in Figure 3.2, samples are highly unbalanced across all 8 known classes. We hypothesise that generating more samples (oversampling) of underrepresented classes through techniques such as data augmentation will significantly improve performance of these classes.

Oversampling is done by combining a different array of techniques each with probability of 0.5 towards a randomly picked sample from a specific class. Therefore, the picked sample which will serve as a basis for augmentation will likely be different each time, depending on the number of samples of the class to be oversampled. Additionally, if a sample is picked more than once, there is a high chance of generating a different sample as the augmentations performed will likely different.

By generating a different sample each time augmentation is performed, we minimize the risk of the model being trained with the same samples over and over again, possibly leading to overfitting.

The augmentations used take advantage of the `Augmentor`[1] library, which provides a wide range of data augmentation techniques, with specific implementations for the tf.keras framework. The following are the ones used for our data augmentation study:

- Horizontal flip
- Vertical flip
- $90^{o}$ rotation
- Shears
- Randomly increase or decrease contrast
- Randomly increase or decrease brightness
- Randomly increase or decrease color intensity
- Tilts from either top to bottom or bottom to top
- Tilts from either left to right or right to left
- Skew image towards a random corner of the image
- Randomly erase a small section of the image
- Distort the original image

These augmentations can either be performed before (offline data augmentation) or during training (online data augmentation) and are applied after taking a central crop of the lesion, in order to augment the lesion itself rather than the surrounding area. Figure 3.5 illustrates examples of these augmentation techniques. One can see that some techniques have a small effect on the original image, while others can significantly change it.

More recent transformations such as Mixup [53] were also considered, but one could argument that such techniques do not make sense in skin lesion classification, as they would combine multiple lesions into a single image, which would not represent a real world scenario and possibly could make the network learn from these potentially misrepresenting features which even an expert human diagnosis would have trouble with.

[1] `https://augmentor.readthedocs.io/en/master/`

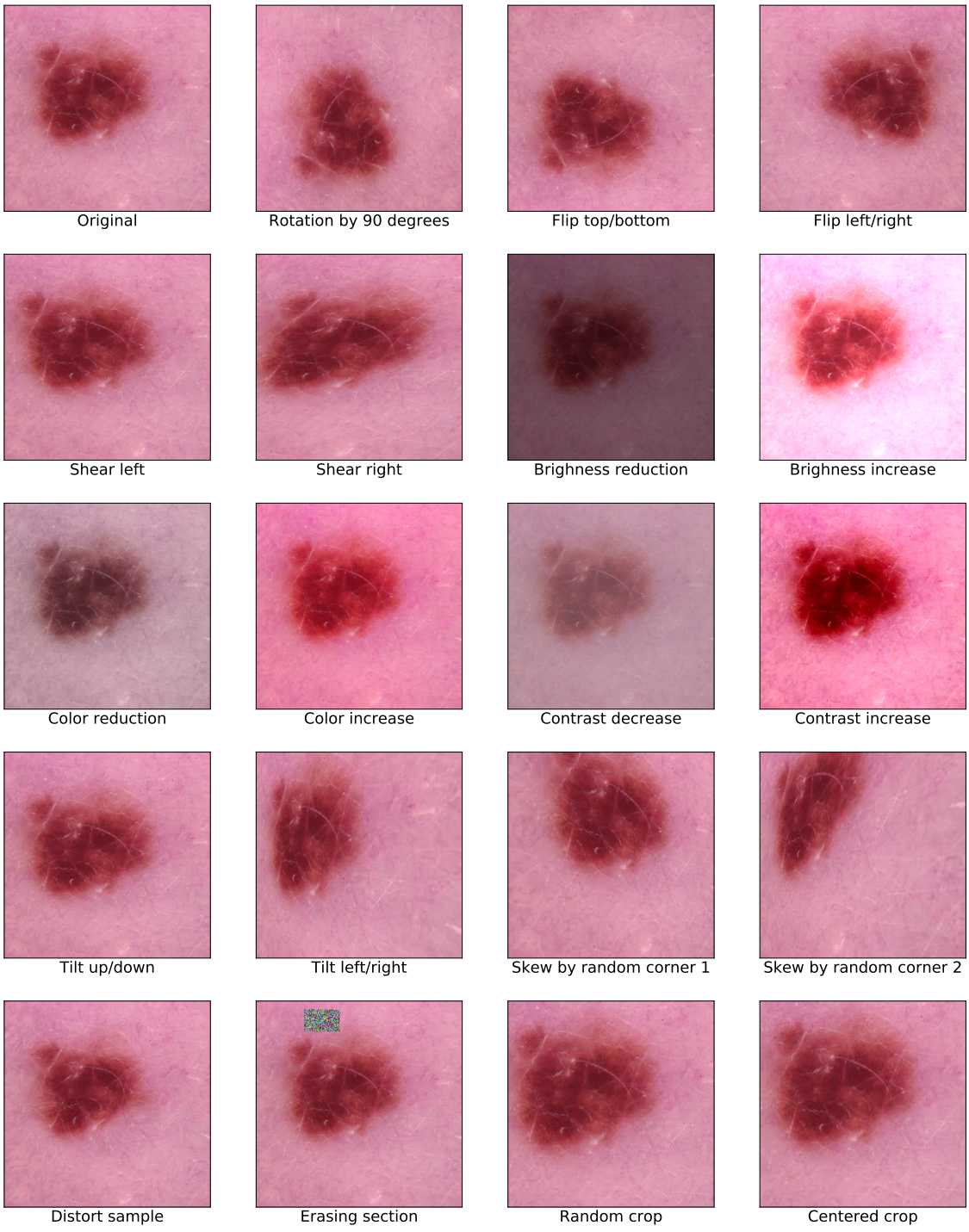


**Figure 3.5:** Examples of augmentations used on chapter 4

### 3.1.5 Split

The number of variables in the future experiments would quickly lead to a combinatorial explosion of configurations, so to minimize the computational cost of the experiments a fixed validation scheme will be used rather than a cross validation scheme. To compensate for this lack of averaging over multiple folds of the data (which gives statistical confidence in

the results), a fixed seed is set for every **PRNG!** (**PRNG!**) as in code snippet 4, which in practice means parameters are initialized identically between experiments (providing some level of statistical confidence when making comparisons) and guarantees reproducibility.

```
def seed():
    from random import seed
    seed(1)
    import numpy.random
    numpy.random.seed(2)
    from tensorflow import set_random_seed
    set_random_seed(3)
```

**Código 4:** Seed function that is called on every experiment to ensure reproducibility and similar conditions between experiments.

The original training and test set from ISIC 2019 is available for direct download, but the test set is not labelled as that information is used internally by the organization for reporting performance with a multitude of metrics. As such, samples from the training set will be split into proprietary training, validation and test sets.

The train with validation and test sets is done is a 80%-20% stratified fashion, which means that 5147 samples from the original set will be part of the test set. Often, other authors from ISIC 2019 use splits with a smaller percentage of the test set, however we want to make sure that our results are not biased towards a specific small test set and as such we chose a bigger one. The test set suffers the same preprocessing steps from the training and validation sets (central crop, resizing and image normalization).

The remaining 80% will be split again into a 80%-20% in a stratified fashion for the train and validation sets, respectively. However, in contrast with the test set these sets will be augmented through offline data augmentation. We believe that by splitting train and test data after the augmentation is performed, one is creating a high bias in the test data as some samples would be variations of samples from the train/validation sets. As such, the test-train split is done before hand from non-augmented samples, as opposed to the train-validation split which happens right before the training routine.

### 3.2 Hardware

The presented training tasks require high computational resources, specially in terms of memory and graphic processing power. As such, it was requested to the the Department of Mechanical Engineering at the University of Aveiro, access to their deep learning research server codenamed Deeplar, that delivers such requirements:

It has four state-of-the-art Graphics Processing Unit (GPU)s along with a high performance Central Processing Unit (CPU) and enough Random Access Memory (RAM) for this work:

- AMD Ryzen™ Threadripper 2950X;
- Four NVIDIA GEFORCE® RTX 2080 Ti;
- 128GB DDR4 RAM.


**Figure 3.6:** Deeplar, the computer used for the presented research

## 3.3 Software

Deeplar runs on a distribution of Linux called openSUSE Tumbleweed 20191004[2]. The common way to interact with NVIDIA GPUs for parallel computing is through an API called CUDA. Deeplar in particular uses CUDA version 10.2 [3]. However, for the task of working with deep neural networks NVIDIA provides a library called cuDNN which allows high level frameworks such as Tensorflow or pyTorch to take advantage of the increased computing power of GPUs. Deeplar uses version 7.6.0 of this library. [4].

For managing the training and testing environment several frameworks are available such as pip, virtualenv or anaconda. The choice for the python environment manager was Miniconda[5], due to it's smaller footprint and ease of use compared with the other mentioned frameworks. The difference between Anaconda and Miniconda lies in the lack of pre installed packages in Miniconda's case. All the code was written in Python 3.6[6] and took advantage of the following packages:

- TensorFlow[7] 2.0.0. Used as a backend of the Keras framework, currently integrated within Tensorflow at `tf.keras`. This allows for training and testing various models through an high level API;
- NumPy[8] 1.15.4 for various vector and matrix operations
- Pandas[9] 1.0.1 for analysing and manipulating with large amounts of structured data.
- Pillow[10] 5.4.1 for image handling and transformations because of this work's image preprocessing needs.

[2] `https://software.opensuse.org/distributions/tumbleweed`
[3] `https://developer.nvidia.com/cuda-zone`
[4] `https://developer.nvidia.com/cudnn`
[5] `https://docs.conda.io/en/latest/miniconda.html`
[6] `https://www.python.org/`
[7] `https://www.tensorflow.org/`
[8] `https://numpy.org/`
[9] `https://pandas.pydata.org/`
[10] `https://pillow.readthedocs.io/en/stable/`

- scikit-learn[11] 0.20.2 for calculating multiple metrics and to split data into train/validation and testing;
- jupyter[12] 1.0.0 for analysing results and creating graphs in a interactive way.
- matplotlib[13] for visualizing results through a multitude of graphs.

The following Github repositories were also used to speed up development:

- Augmentor[14] 0.2.8. Used as a image augmentation library that provides a wide range of simple and complex augmentation operations.
- EfficientNet Keras[15] 1.15.4. This is a open source implementation of EfficientNets for the Keras framework. Tesnorflow 2.0.0 does not provide a implementation of EfficientNet, however it is schedules for future releases to be integrated within the tf.keras framework.

[11] https://scikit-learn.org/
[12] https://jupyter.org/
[13] https://matplotlib.org/
[14] https://www.tensorflow.org/
[15] https://numpy.org/

CHAPTER 4

# Experiments

This chapter aims to describe the experiments in terms of reproducible steps, settings, parameters, conditions, as well as present and discuss results. The ultimate goal is to create an approach for the first task of the ISIC 2019 challenge in such a way that conclusions can be drawn from it.

Several important aspects were identified and talked about in chapter 2 that are essential for such an approach. Therefore, this chapter is divided into 5 different sections which hopefully addresses each of those aspects in a systematic way.

First, section 4.1 will study which pre-trained models work best for this classification task and the overall impact of transfer learning and neural network architectures in problems like these. Next, section 4.2 will methodically optimize the chosen model's hyperparameters through common reasoning to hopefully improve performance and draw conclusions from it. This will be followed by section 4.3 which will address different methods of dealing with unbalanced data and discuss the importance behind such task. Then, section 4.4 will improve the approach's performance by finding a suitable way for ensembling multiple models. Finally, section 4.5 dealing with out of distribution through different procedures.

## 4.1 Pre-trained model choice

In order to classify skin lesions through deep neural networks two approaches can be taken. The first is to create a model from scratch, design it's architecture and train it from the ground up. However, this approach is troublesome because it requires reasoning, setting many hyperparameters simultaneously and cross-validating a wide range of values which is computationally expensive. Another approach is through the use of transfer learning by leveraging the weights of pre-trained models. This is a good option when data is scarce, which is often the case of computer vision related problems. Additionally, there is a wide range of pre-trained models to chose because of challenges such as such as the ILSVRC.

In 2 several pre-trained models were shown such as ResNet, DenseNet, VGGNet and EfficientNet. All of these were pre-trained on ImageNet with millions of samples of photos of

categories such as dogs or cats. As it stands, these models can be re-purposed for skin lesion classification through transfer learning, which was the most common approach to the ISIC 2019.

The first step is to filter which pre-trained models should be considered for the problem of skin lesion classification. For our study we consider commonly used pre-trained models from skin lesion classifiers presented in chapter 2, which are readily available through the tf.keras.applications framework. More specifically, different variations from the VGG, ResNet, DenseNet, Inception and EfficientNet pre-trained models were considered. Each of these have their own architectural principles and design concepts, and their corresponding variations are based on a baseline model which is scaled up and down.

A benchmark has been set in order to provide a fair evaluation for each of these models:

- Each model is trained on a undersampled version (4000 training samples and 1000 validation samples) of the ISIC 2019 dataset that maintains the original class distribution. The undersampling process is done by randomly picking samples from the original dataset in a stratified manner. This smaller dataset will substantially decrease each model train time, which will allow us to train more pre-trained models in a agile manner. Presumably, a undersampled version of the original dataset will yield similar results as if the whole dataset was being used.
- All image samples are standarized relative to the ISIC 2019 dataset mean RGB channel values.
- Extracting all layer's weights while also fine-tuning these layers, which presumably yields higher performance since it will continue to optimize parameters relative to the target dataset, thus minimizing error on said dataset and generalizing well to similar test data. Several other approaches were also tried such as only fine tuning from the middle layer to the top of the network (i.e.) or freezing all the network and only training the classifier. However, both had considerably worse performance, and possibly required a more comprehensive study like the one done in [54] for each of the pre-trained models which would be troublesome.
- The top layer from each pre-trained model is removed and replaced by a global average pooling layer to reduce the number of parameters before the classifier.
- The classifier is composed of one layer of 512 neurons which are fully connected, and each neuron uses the ReLU activation function and one softmax layer with 8 neurons to translate each of the class's probabilities. Meaning that the model classification is given by the softmax value which is the highest. In Figure 4.1 one can see 3 examples of samples taken from ISIC 2019 dataset being classified into probabilities by one of the models.
- Online data augmentation is performed with random crops, flips, rotations, shears, brightness changes and color changes, in order to minimize overfitting while training.
- All image samples are resized before passing through the network to each pre-trained models input size. All the presented pre-trained models have a square input aspect ratio, so no image distortions happen while resizing.

- The following the work done by [38] the Adam optimizer is used, with a categorical cross entropy loss function.
- A batch size of 32 is used with a learning rate of 0.0001. However, the learning rate is reduced by a factor of 0.1 when validation loss stops improving for 8 epochs.
- Train for a maximum of 100 epochs. However, early stopping is performed whenever validation loss stops improving for 16 epochs.
- For each training process 3 models are saved. The model up to a certain epoch that obtained the highest balanced multi-class accuracy on the validation set, the model up to a certain epoch that obtained the lowest loss on the validation set, and finally the latest model from the last epoch. It is hypothesized that the latest model will have worse generalization than that of the other 2 models.

The simplest way of re-purposing a pre-trained model is by only replacing it's classifier. For this procedure, the top layers of the pre-trained model must be removed, which are the layers that are not convolutional or pooling layers at the end of the pre-trained model's architecture, and the weights from the trainable layers of the convolutional base must be freezed, by setting their trainable property to false. Figure 4.2 compares the results of applying this approach to several pre-trained models in terms of test accuracy and test balanced multi-class accuracy. One can see that the overall performance is bad, with the best architectures being VGG and EfficientNet. These results are to be expected because the weights of the layers of the convolutional base of these models were all trained to a very different dataset (ImageNet), from the one we are now trying to re-purpose the model to (ISIC 2019 dataset).

Therefore, a more sensible approach to this problem is to adapt the whole set of parameters to this new dataset while also taking advantage of the already existing weights trained on ImageNet. This approach is based on the fine-tuning concept, meaning that all the weights from the pre-trained model are transferred to the new model, but are also updated along with the classifier on each epoch. Presumably, by taking this approach, the knowledge obtained from the existing weights will be adapted to a new problem and increase performance results on the test set. Figure 4.3 shows that this assumption is true, as all pre-trained model's had a significant increase in both accuracy and balanced multi-class accuracy when compared with 4.2.

In general, one can see that the balanced model accuracy is much lower than the accuracy in every trained model. That is due to the fact that the data is imbalanced and more true predictions are made towards classes with higher number of samples (illustrated in Figure 4.4), which the accuracy metric does not take into account. On the other hand the balanced multi-class accuracy is the mean recall of each class and therefore the performance of the model in each class is weighted the same, which substantially lowers the values.

Generally, by looking at Table 2.1 we can see that more recent architectures such as DenseNet or Inception perform better than older architectures such as VGG. This can be attributed to the fact that the VGG16 and the VGG19 are shallower models compared to the others. It is to be expected that deeper models have a positive impact on model accuracy,

because more layers have the ability to create more levels of abstraction. To put this into test we compared for each "family" of architectures the influence of depth on model accuracy.

The one exception to the depth rule is presented by the VGG family of models, which does not improve performance with depth increase (Illustrated in 4.5). In fact, VGG19 also performed worse than VGG16 on the ILRSVC challenge as shown in 2.1. We presume this is related to the vanishing gradients problem, which typically becomes a larger concern for deeper networks because the gradient decreases exponentially as we propagate down to the initial layers [6].

This problem was later addressed by architectures such as ResNet, which would allow it to build models up to 152 layers deep. In this study we will compare ResNet50, ResNet101 and ResNet152 with 50, 101 and 152 depths, respectively. We can see the impact of model depth on ResNet performance in 4.6, which shows that model depth is directly correlated to validation balanced accuracy in a linear way for this specific architecture. He et al [9] tried to go further and created a ResNet1202 layer deep network, but it performed worse than the 152 version, which it was resolved as an overfitting problem due to a large number of parameters for the size of the given dataset, showing the disadvantages of models which are too large.

DenseNet also extended the idea further which would allow it to build even deeper models, namely with 121, 169 and 201 layers. It should also be noted that there is a 264 version of this network, however as it was not available for the tf.keras framework it was discarded. One can can see how depth impacts the validation performance of the DenseNet architecture on 4.7. Surprisingly, DenseNet201 has the worst performance on the training set, however it also has the highest BMA, contrarily to DenseNet169 which is overfitting due to the huge disparity between training BMA and validation BMA.

Even though within the same architecture deeper models tend to perform better, the increased depth means more trainable parameters which means that the requirements for memory and processing power increase. An inefficient use of model parameters might lead to overfitting because the network can tune them to a very specific set of data **?**. EfficientNets address this problem with the compound scaling method presented in chapter 2, which not only scales depth, but also input input size and layer width. We can see it's effectiveness in 4.8.

As shown in the confusion matrix of Figure 4.9, even the DenseNet201, which was the best performing model, struggles on classification of underrepresented classes such as vascular lesions, while performing well with overepresented classes such as the melanocytic nevus. Considering the results, more data from underrepresented classes need to be fed into the training process in order to improve classification performance, Nonetheless, this provides the baseline model which should serve as a benchmark comparison in the next sections.

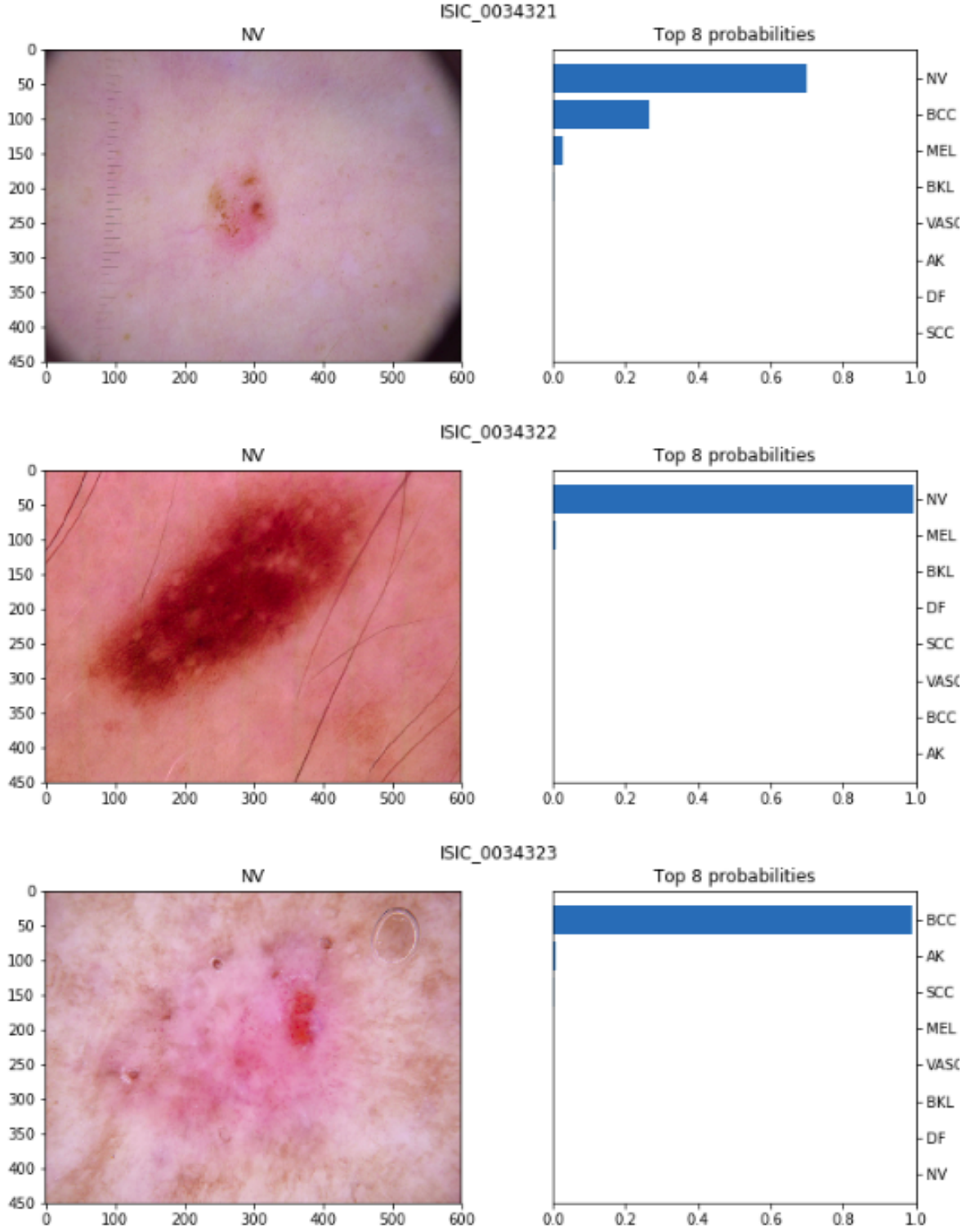


**Figure 4.1:** 3 Softmax probabilities created from 3 samples of the ISIC 2019 dataset

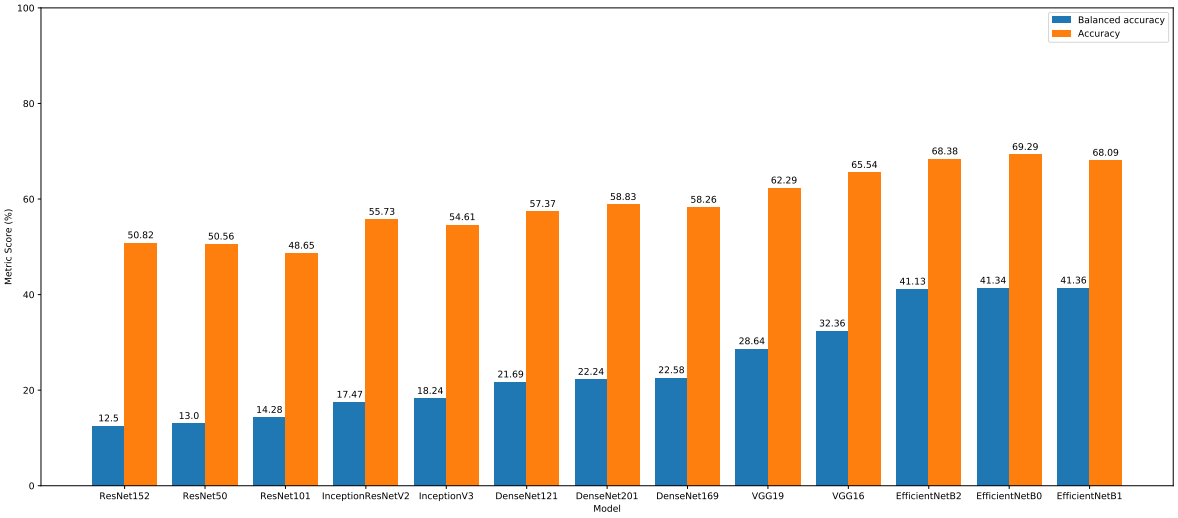


**Figure 4.2:** Balanced multi-class accuracy and accuracy of various pre-trained models by replacing the top layers (classifier)

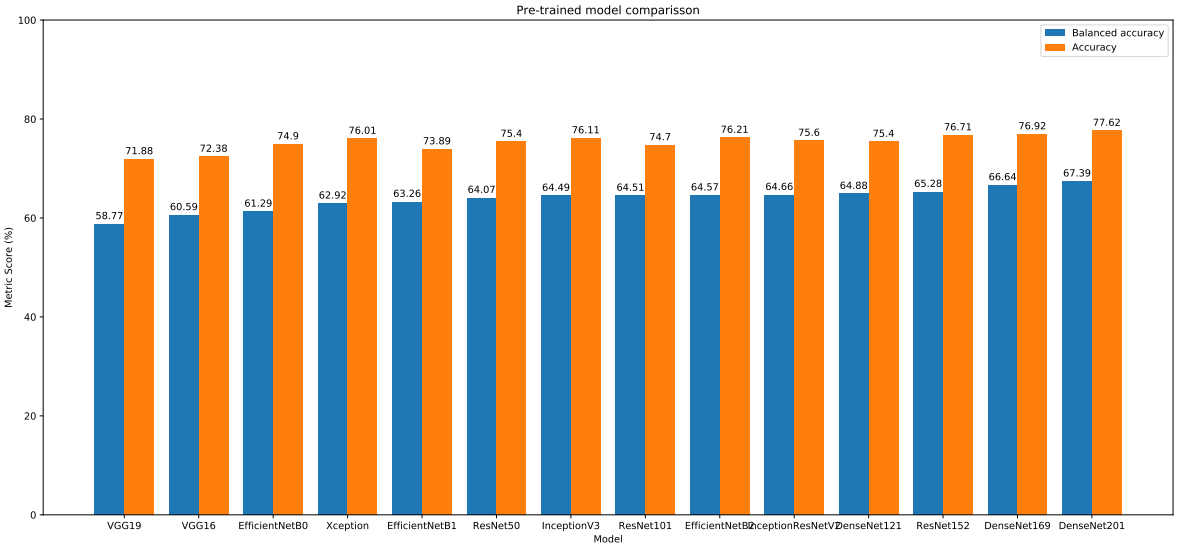


**Figure 4.3:** Balanced multi-class accuracy and accuracy of various pre-trained models on fine tuning the whole model and replacing the top layers(classifier)

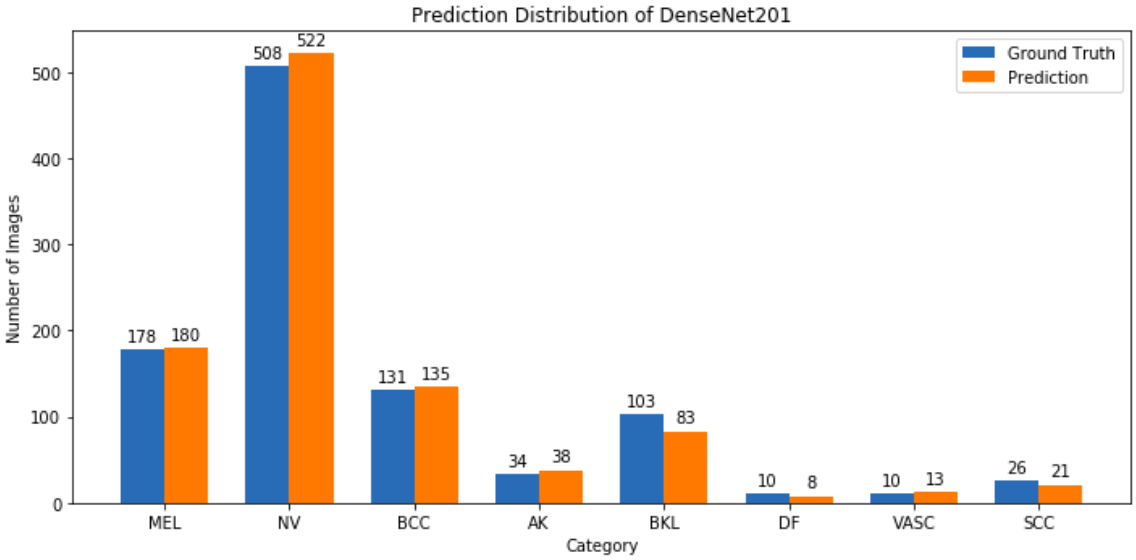


**Figure 4.4:** Prediction distribution of DenseNet201

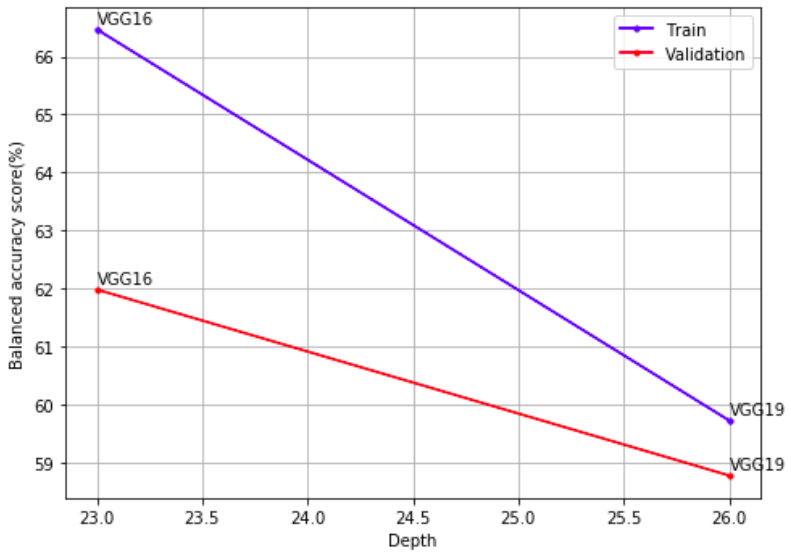


**Figure 4.5:** Influence of depth on VGG pre-trained models

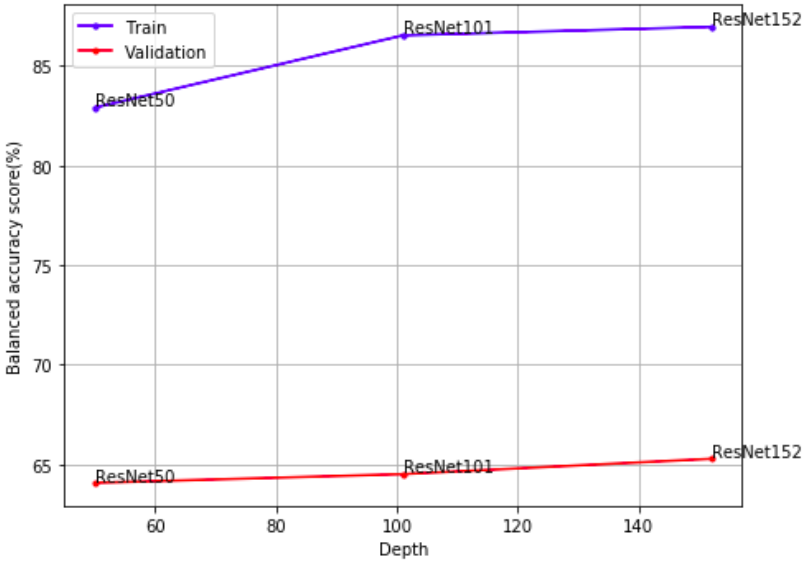


**Figure 4.6:** Influence of depth on ResNet pre-trained models

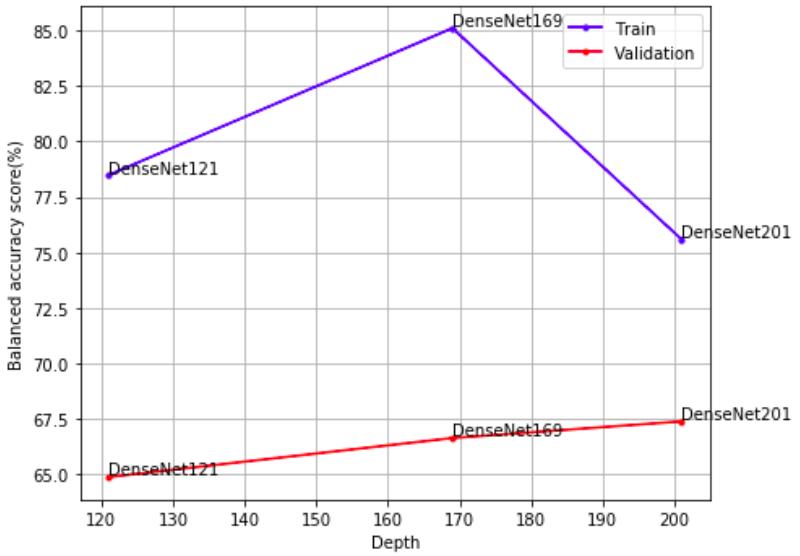


**Figure 4.7:** Influence of depth on DenseNet pre-trained models

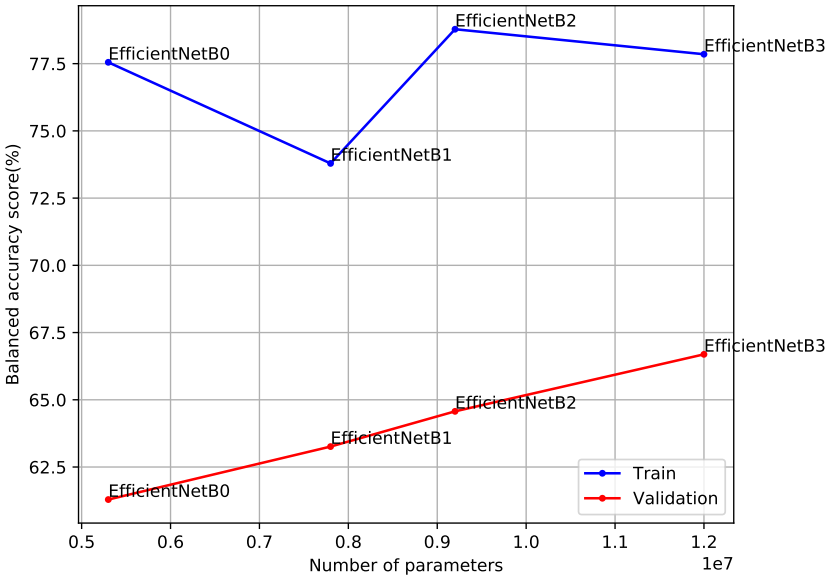


**Figure 4.8:** Influence of model parameters on EfficientNet performance


**Figure 4.9:** Confusion matrix of the hyperparameter tuned DenseNet201

## 4.2 Hyperparameter optimization

In order to achieve the best possible performance using a transfer learning method, pre-trained models need to be hyperparameter tuned to obtain the best possible combination of hyperparameters for the problem at hand. As such, in order to tune hyperparameters usually the holdout method is applied, meaning that a different set of data (validation set) is used for evaluation to get an unbiased assessment of the model's generalization performance. The validation set is required because if one uses the test set for this optimization then there is a high risk of optimistically biasing the model [55]. Even though the holdout method has it's downsides, we applied it because time requirements are much lower compared to techniques like nested cross validation or k-fold cross validation.

In order to systematically find each hyperparameter in an efficient way, the strategy presented in **??** will be used. Additionally, for this evaluation a benchmark has been set in order to compare the results in a fair manner:

- Each model is trained on a undersampled version (4000 training samples and 1000 validation samples) of the ISIC 2019 dataset that maintains the original class distribution. The undersampling process is done by randomly picking samples from the original dataset in a stratified manner. This smaller dataset will substantially decrease each model train time, which will allow us to train more pre-trained models in a agile manner. Presumably, a undersampled version of the original dataset will yield similar results as if the whole dataset was being used.
- All image samples are standarized relative to the ISIC 2019 dataset mean RGB channel values.
- Extracting all layer's weights while also fine-tuning these layers, which presumably yields higher performance since it will continue to optimize parameters relative to the target dataset, thus minimizing error on said dataset and generalizing well to similar test data. Several other approaches were also tried such as only fine tuning from the middle layer to the top of the network (i.e.) or freezing all the network and only training the classifier. However, both had considerably worse performance, and possibly required a more comprehensive study like the one done in [54] for each of the pre-trained models which would be troublesome.
- The top layer from each pre-trained model is removed and replaced by a global average pooling layer to reduce the number of parameters before the classifier.
- The classifier is composed of one layer of 512 neurons which are fully connected, and each neuron uses the ReLU activation function and one softmax layer with 8 neurons to translate each of the class's probabilities. Meaning that the model classification is given by the softmax value which is the highest. In Figure 4.1 one can see 3 examples of samples taken from ISIC 2019 dataset being classified into probabilities by one of the models.
- Online data augmentation is performed with random crops, flips, rotations, shears, brightness changes and color changes, in order to minimize overfitting while training.

- All image samples are resized before passing through the network to each pre-trained models input size. All the presented pre-trained models have a square input aspect ratio, so no image distortions happen while resizing.
- For each training process 3 models are saved. The model up to a certain epoch that obtained the highest balanced multi-class accuracy on the validation set, the model up to a certain epoch that obtained the lowest loss on the validation set, and finally the latest model from the last epoch. It is hypothesized that the latest model will have worse generalization than that of the other 2 models.

One important aspect of training deep neural networks is the number of epochs that it will run on. One epoch is when all training examples have been forward and backward passed through the network. In other words, one epoch means that every image has been seen. Therefore, the number of epochs define how many times each image is seen by the network. We fixed the maximum number of epochs to 100 (even though most of our models never reach close to it).

However, we are fine tuning a pre-trained model with a new classifier on top, which means that the classifier's weights are all at 0. We hypothesized that the initial weights of the classifier are determinant for the overall model performance. So in order to deal with this problem for an initial number of epochs we will freeze all the pre-trained model's layers and only tune the classifier weights just to minimize the impact of the initial weights of the network on the overall performance. Figure 4.10 illustrates the impact of the number of weight initializing epochs on the overall performance.

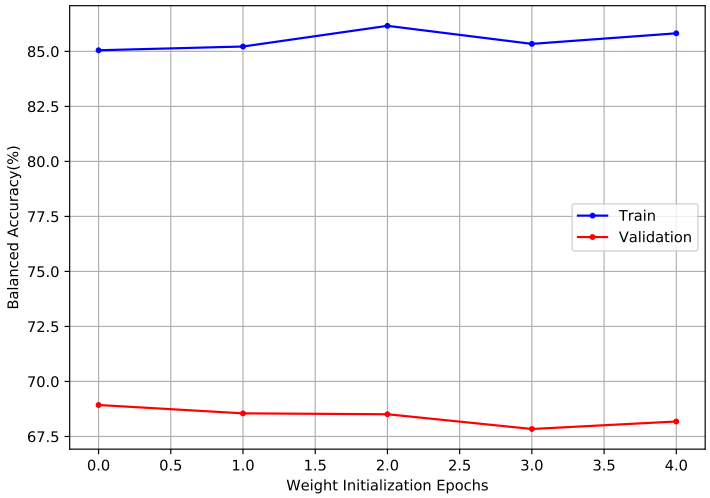


**Figure 4.10:** Influence of weight initialization epochs on train and validation balanced multi-class accuracy

Batch size can also be an important factor to consider. It defines the number of training images passed through the network in one forward and backward pass. The bigger the batch size the more memory is needed and the faster the network is going to converge, because there are fewer iterations per epoch. The size is directly correlated to the number of iterations per

epoch by equation 4.1.

$$\text{iterations per epoch} = \frac{\text{training images}}{\text{batch size}} \tag{4.1}$$

Where one iteration is one backward and forward pass, which means that larger batch sizes will make less iterations per epoch, which means that the model will train faster.

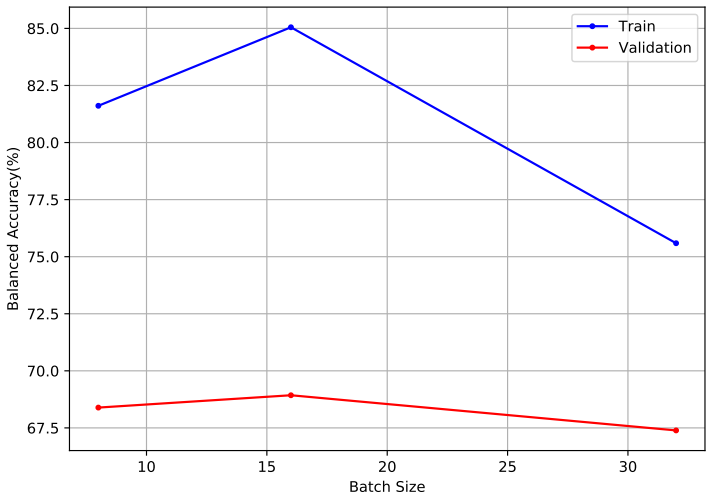


**Figure 4.11:** Influence of batch size on train and validation balanced multi-class accuracy

Perhaps the most important hyperparameter to optimize is the learning rate. It controls the step size for model weight updates with respect to the loss function. Lower values mean slower travel along the downward slope, which consequently takes longer times to converge. In this approach, two learning rates are considered.

We vary the learning rate which is used to initialize the weights of the classifier during weight initialization epochs and illustrate the results in figure 4.12.

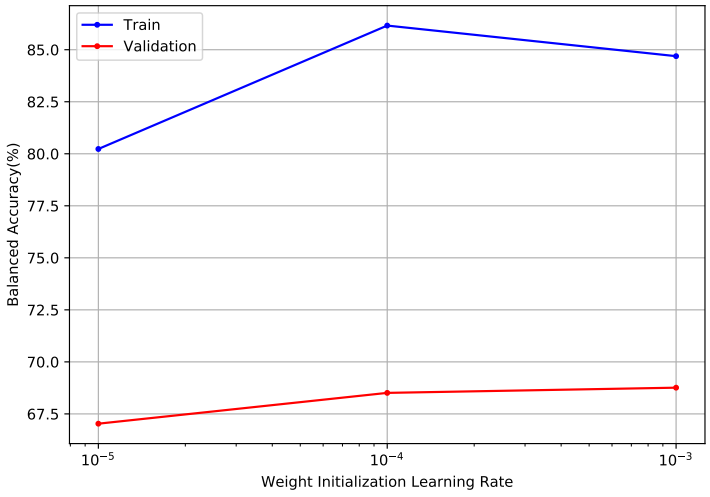


**Figure 4.12:** Influence of weight initialization learning rate on train and validation balanced multi-class accuracy

Figure 4.13 illustrates the evolution of the learning rate over the epochs for different weight initialization learning rates. One can see the lower learning rates make the model to converge

later and specifically with a learning rate of $1e-5$ the model converges to a local minimum earlier on.

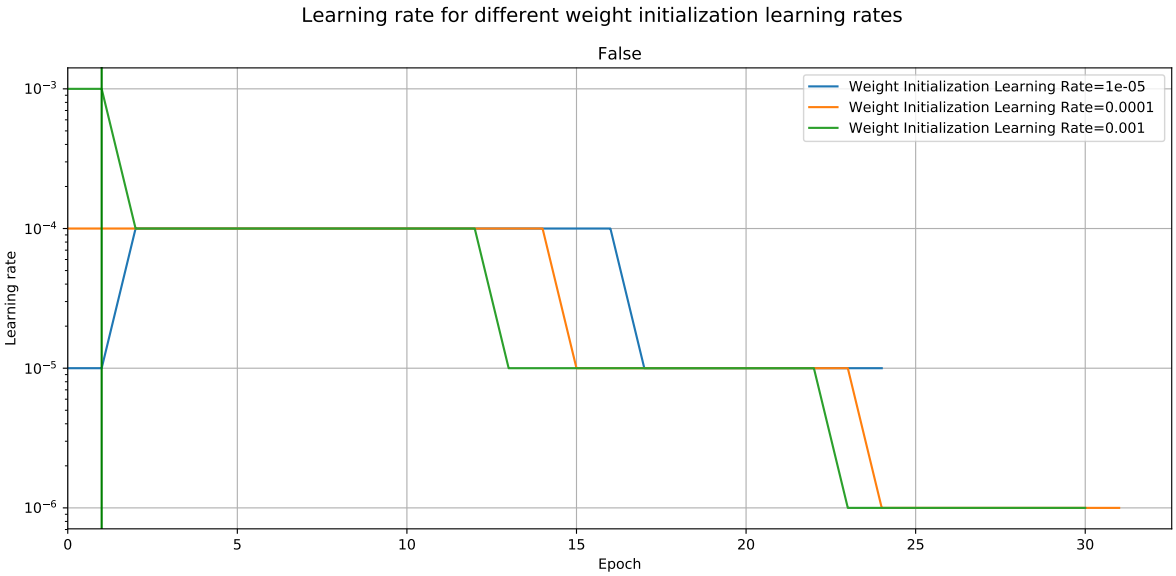


**Figure 4.13:** Influence of weight initialization learning rate on the evolution of learning rate over epochs

The most impactful learning rate is the start fine tuning learning rate which is used to fine tune the whole model after the weights of the classifier had been initialized. Hypothetically, choosing too small learning rate values will result in a long training process that could get stuck in a local minimum, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. We verify this hypothesis by comparing various fine tuning learning rates, which are compared in Figure 4.14.

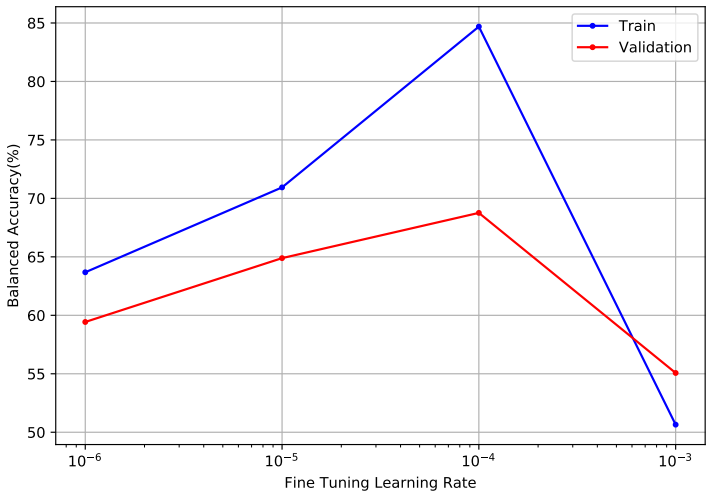


**Figure 4.14:** Influence of fine tuning learning rate on train and validation balanced multi-class accuracy

Figure 4.15 shows that a low learning rate like $1^{-6}$ make the improvements linear, while higher learning rates like $1^{-4}$ tend to show a more exponential curve because they will decay

the loss faster. However, when the learning rates are too big they can get stuck at worse values of loss which is the case of $1^{-3}$. This is because there is too much "energy" in the optimization and the parameters are bouncing around chaotically, unable to settle in a nice spot in the optimization landscape.

The learning rate of $1^{(}-4)$ seems to be the optimal choice because it allows the model to have a higher chance of finding a better region of search space in the initial epochs, which consequently influences the rest of the training process because the rest of the epochs will be spent on minimizing the loss within that specific region, that is finding the local minimum. In contrast, learning rates like $1^{-5}$ and $1^{-6}$ have a lower chance of finding a good local minimum at the beginning of the training process and therefore take more epochs to converge, and higher learning rates like $1^{-3}$ are too big such that the network never finds a good region of search space at the beginning of the training process, which influences the rest of the training because the latter lower learning rates will try to optimize the loss within a bad search space.

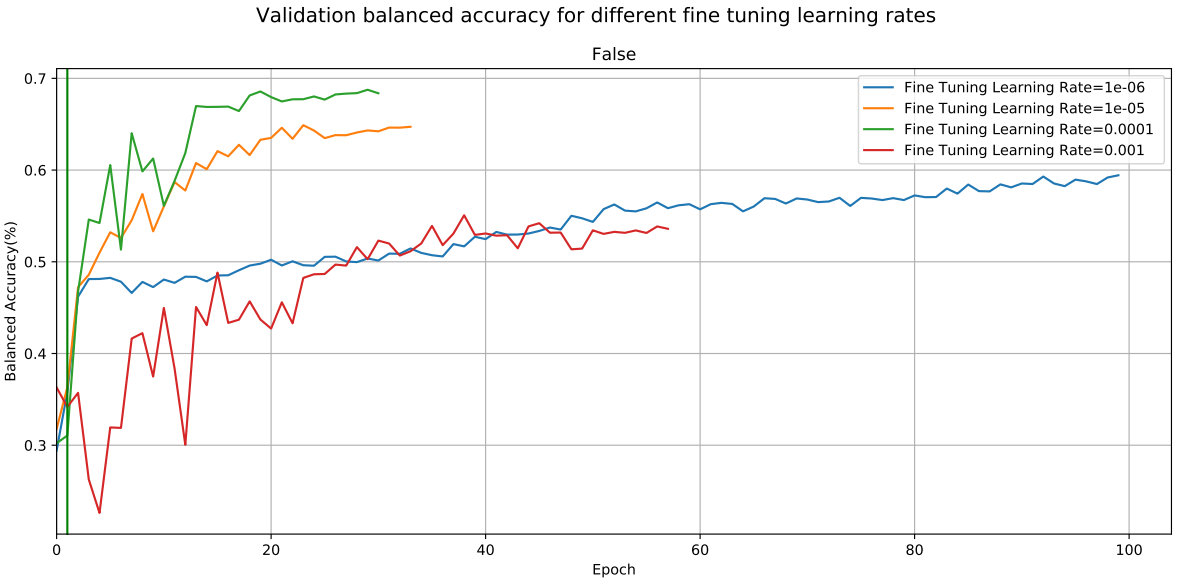


**Figure 4.15:** Influence of fine tuning learning rate on train and validation balanced multi-class accuracy over epochs

From in 4.14), one can see that even for the chosen hyperparameters the model still suffers from overfitting, as seen by the huge discrepancy between training and validation balanced multi-class accuracy. In light of the problem, l2 regularization was used, which adds a cost to the loss function of the network for large weights (or parameter values), and presumably result in a overall simpler model that will be forced to learn only the relevant patterns in the train data. However, results (see 4.17) show worse overall performance. Other techniques such as dropout were also experimented with (see 4.18), but again no significant improvements were to be shown.

DenseNet201 has a considerably large number of trainable parameters when compared with other models such as (see 2.1), which in theory means that it is easier for the model to memorize samples, and therefore overfit. As such, by lowering the capacity of the network, one will force it to learn the patterns that matter or that minimize the loss. To put this into

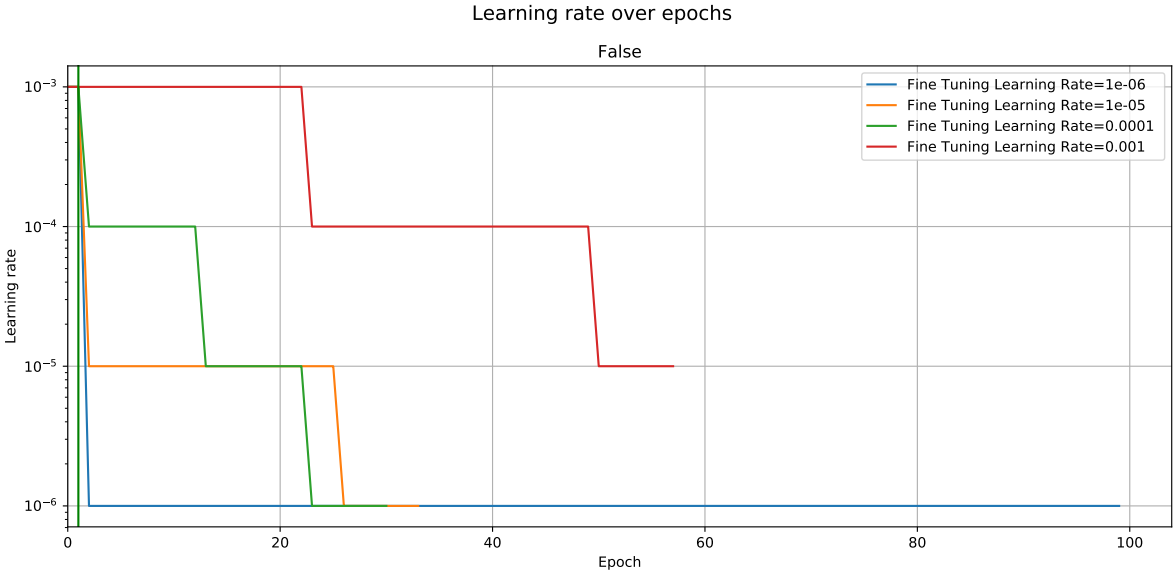


**Figure 4.16:** Influence of fine tuning learning rate on the evolution of learning rate over epochs

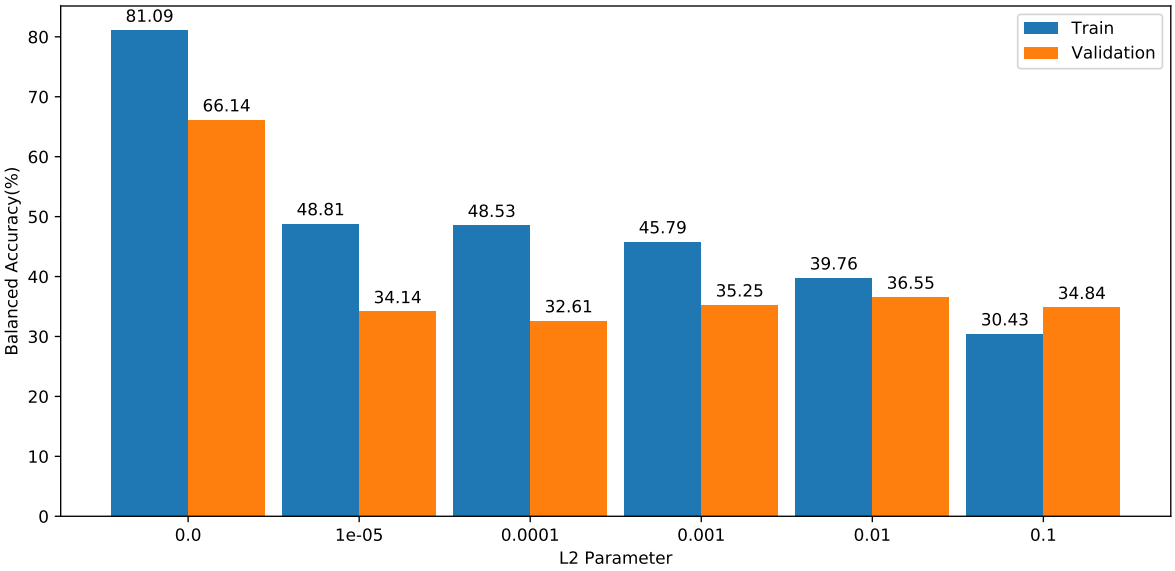


**Figure 4.17:** Influence of l2 regularization parameter on train and validation balanced multi-class accuracy

test, results for variations with a smaller number of parameters of the DenseNet, specifically, DenseNet121 and DenseNet169, were compared with DenseNet201 in Figure 4.19. Results show that indeed lowering the network's capacity improves performance, but not by a significant margin.

With the parameters now set the model is now able to get accuracy of 75.4% and a balanced multi-class accuracy of 59.1%. The values of the balanced multi-class accuracy are low relative to the accuracy because some classes have poor performance in comparison with other classes (this effect is shown in the confusion matrix of Figure **??**), which the accuracy metric does not take into account. One can. Overall, the attained performance values so far are relatively low.

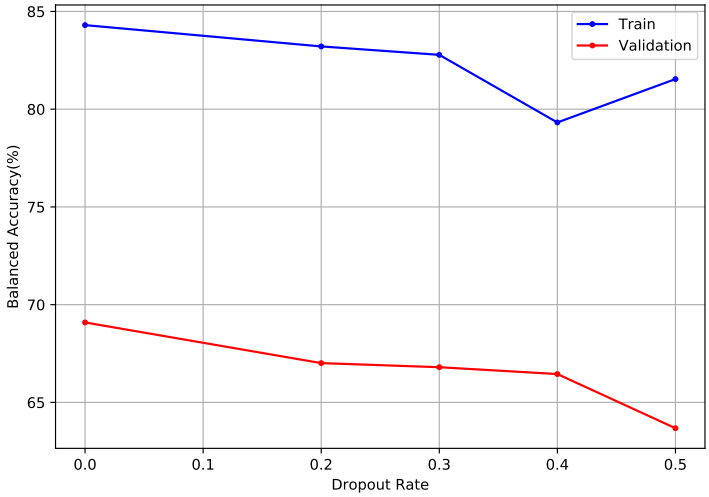


**Figure 4.18:** Influence of dropout rate on train and validation balanced multi-class accuracy

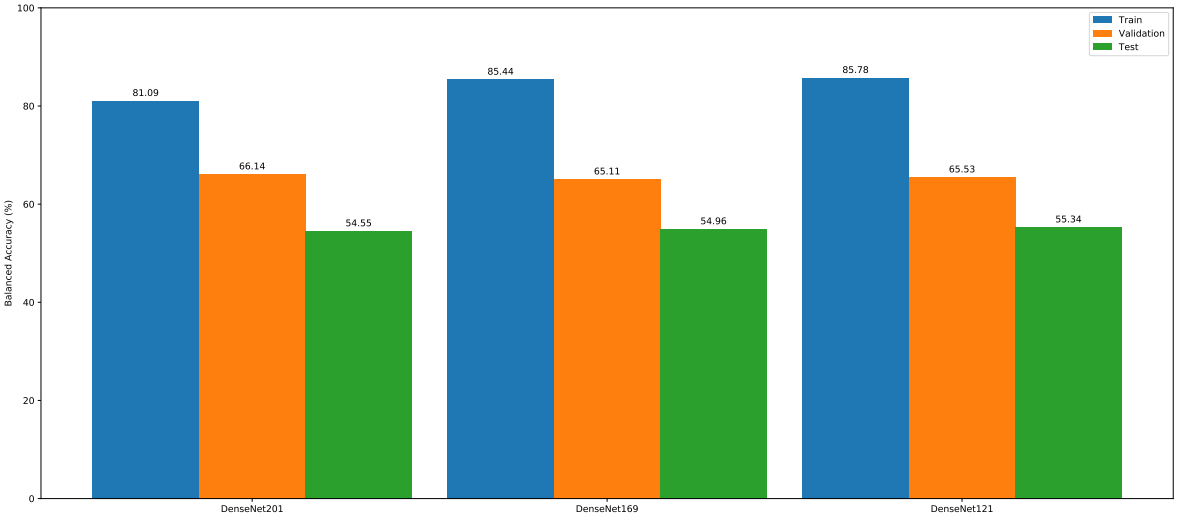


**Figure 4.19:** Comparison of different variations of DenseNet on train, validation and test balanced multi-class accuracy for a dataset with 5000 unbalanced samples


**Figure 4.20:** Confusion matrix of the hyperparameter tuned DenseNet201

The validation curve of the hyperparameter tuned model in figure **??** shows a considerable amount of overfitting, as the $BMA_{validation}$ does not approximate $BMAtrain$. Presumably, these results stem from an inappropriate number of training samples relative to the number of free parameters in the network. The dataset used so far has been an undersampled version of the ISIC 2019 challenge dataset, specifically with 4000 training samples and 1000 validation samples that keep the same class distribution as the original ISIC 2019 challenge dataset. This means that underrepresented classes like dermatofibroma or vascular lesion (see Figure 3.2) have too few samples to learn strong generalizable features. Indeed, an increase in dataset size enables the DenseNet201 model to overfit less and attain better $BMA_{validation}$ values (see 4.21), which corroborates the importance of data quantity for training deep neural networks.

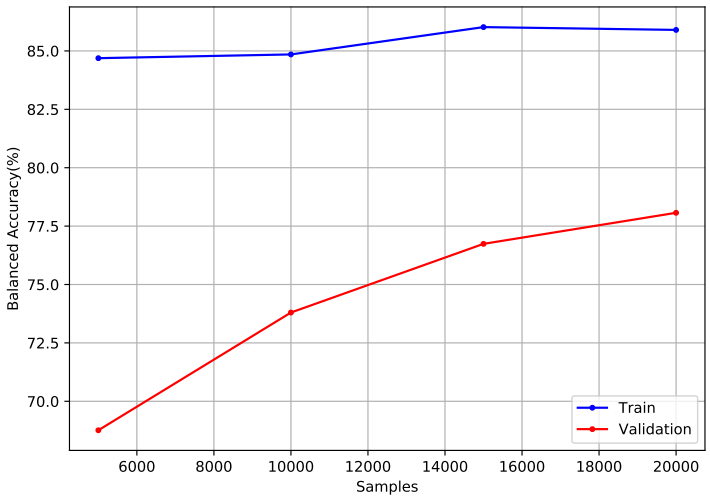


**Figure 4.21:** Influence of the number of samples on the best values of balanced multi-class accuracy of train and validation sets

The results from the plot graph in Figure 4.19 showed that one could use a smaller pre-trained model from DenseNet, while maintaining similar performance, which would overall be benefitial to reduce overfitting. However, with a bigger dataset one could argue that the gap between models with high and low number of trainable parameters would increase. but that is still not the case (see Figure 4.22). Even though DenseNet201 got the best performance in comparison with it's smaller versions, this improvement is unsubstantial as the difference between the DenseNet121 and DenseNet201 on test data is about ?%. Considering the results, that difference becomes even more irrelevant when taken into consideration that the gap between train and validation on DenseNet121 is substancially smaller than the one of DenseNet201. As such, a smaller model proves to be a better choice as it reduces overfitting while keeping similar performance to bigger models.

From Figure 4.23, one can see that training with more samples takes longer, which is mainly due to the learning rate scheduler. In models with more samples, validation loss keeps improving for longer for higher learning rates, which makes the scheduler decrease the learning rate later compared to models trained with less samples (see Figure 4.24). Consequently, as

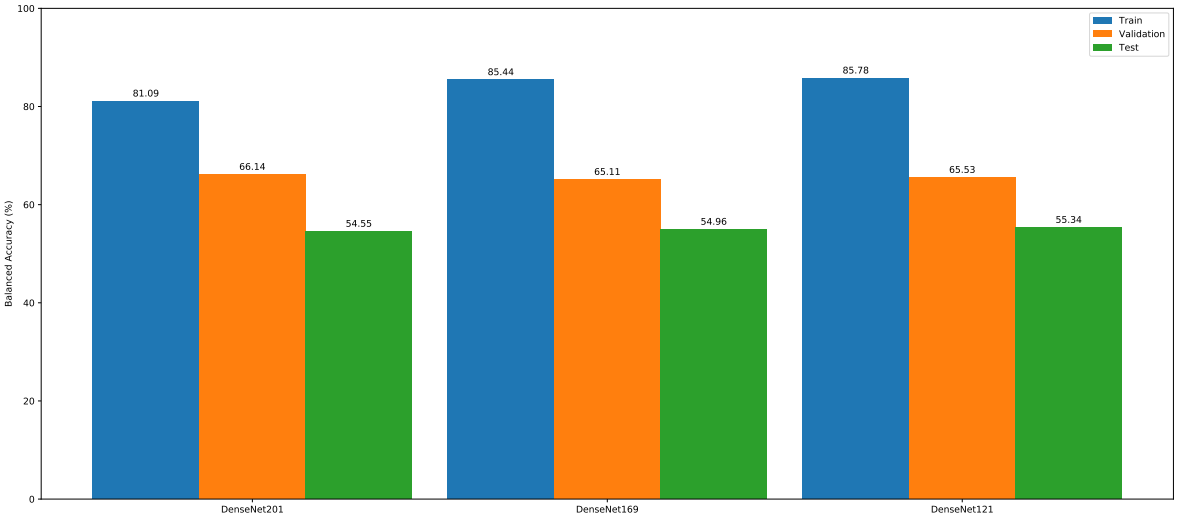


**Figure 4.22:** Comparison of different variations of DenseNet on train, validation and test balanced multi-class accuracy for a dataset with 20264 unbalanced samples

most models converge to local minimums with low learning rates, models trained with more samples take longer to converge because they take longer to reach low learning rates.

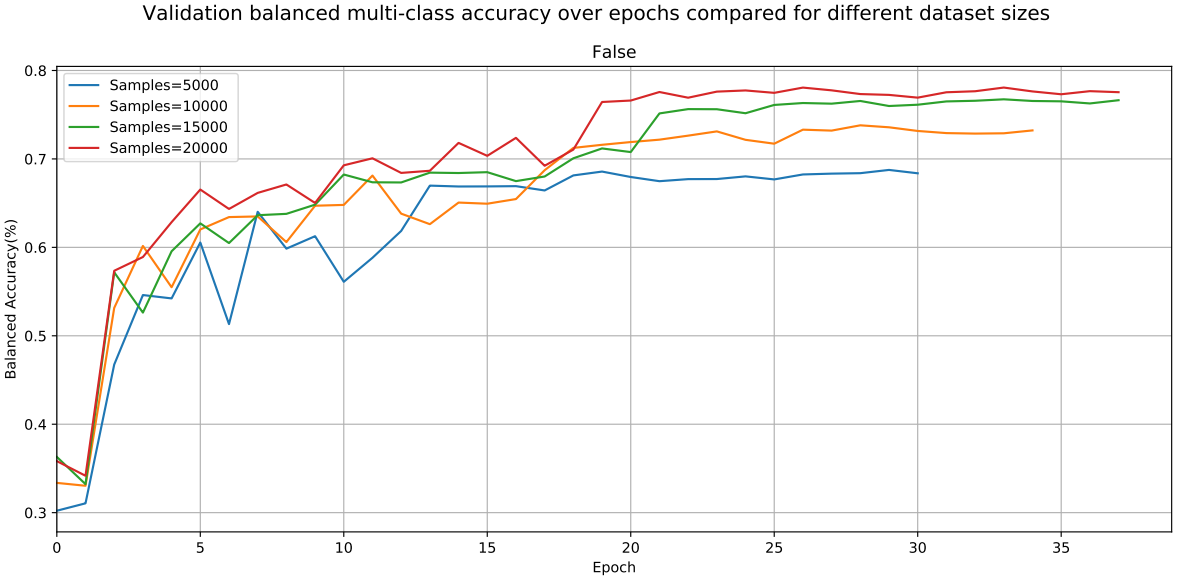


**Figure 4.23:** Influence of the number of samples on validation balanced multi-class accuracy over epochs

Considering the results from 4.21, there is a considerable improvement on the overall performance by increasing the dataset. Virtually one could assume that increasing the dataset size through other techniques such as data augmentation could improve the overall performance of the network, specially in underrepresented classes. Different strategies of data augmentation can be used, one of them being offline data augmentation, meaning that a bigger dataset is generated before the training process from an original dataset through augmentation techniques. However, there are different methods for augmenting images such as rotations or distortions and they can be combined together to produce images which are substancially

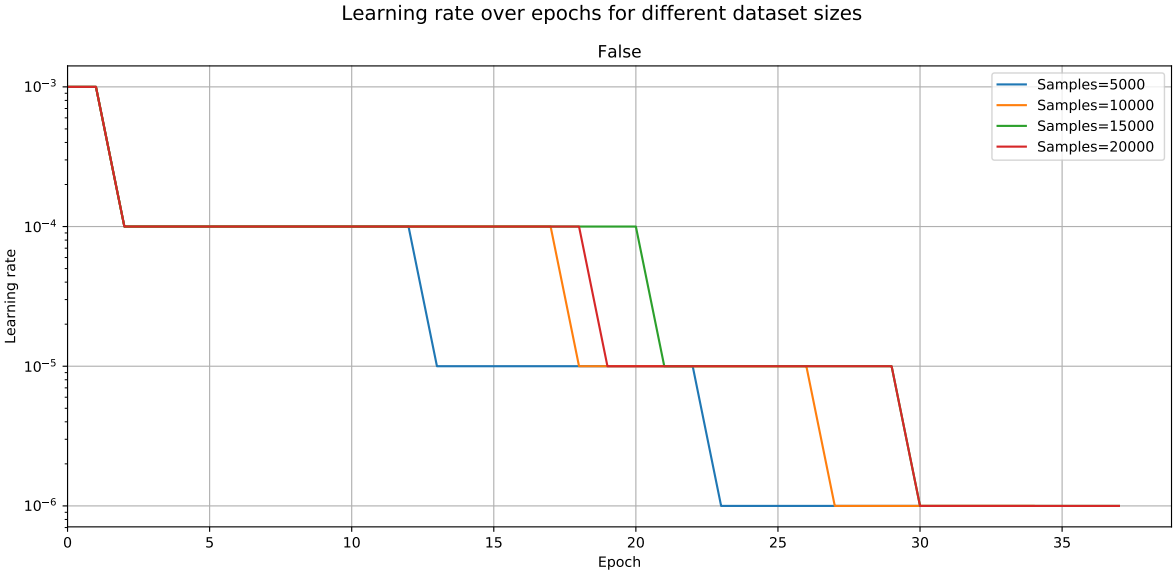


**Figure 4.24:** Influence of the number of samples on the learning rate over epochs

different from their originals. As such, depending on the number of augmentations to be considered there could be a huge amount of different combinations between them specially considering that some of them have different variations and parameters which can quite change the augmentation (the magnitude of a distortion or the angle of a rotation). Therefore, it is important to determine what combinations of data augmentation techniques significantly improve the overall performance for the problem of skin lesion classification.

Figure 4.25 shows a comparison between different augmentation groups applied to the hyperparameter tuned DenseNet201 obtained from section 4.2 with 20264 class balanced samples (2533 per class), meaning that underrepresented classes are oversampled through the augmentation techniques from each of the groups, and overrepresented classes are undersampled by choosing samples randomly from that class without repetition. In turn, this means that only ?????x samples are oversampled and the rest of the images are original ones. The presented groups are:

- Group 0: No augmentation techniques are applied. Oversampling is done by randomly copy and pasting samples from a specific class with repetition. This is the baseline which other groups should be compared against.
- Group 1: Composed of rotations of 90 degrees, flips from top to bottom or vice versa, flips from left to right or vice versa, not centered crops that keep 85% of the original size and shears with a variation anywhere from 20 degrees towards the left side of the image to 20 degrees towards the right side of the image.
- Group 2: Composed of the augmentation techniques of group 1 plus some adjustments on pixel intensities, specifically, brightness changes from 50% to 150% (100% being the original image and 0% being a black image), contrast changes from 50% to 150% (100% being the original image and 0% being a solid grey image), and coloration changes from 50% to 150% (100% being the original and 0% being a black and white image).
- Group 3: Composed of the augmentation techniques of group 1 plus perspective trans-

formations. These transformations include tilts on either the left or right side, tilts forward and backward and skews from a random corner of an image.

- Group 4: Composed of the augmentation techniques of group 1 plus noise induction augmentation techniques. Elastic distortions are applied with a grid size of 8, meaning that the image is divided into 8x8 cubes and each region is distorted independently. Random erasing is also applied, which takes a random area equivalent to 25% of the original image and replaces it by random pixels, meaning that each pixel has a random intensity producing noise in that area.

In group 1, 2, 3 and 4 there is a probability of 0.5 of applying a specific augmentation technique in the group augmentation pipeline, which means that the same sample augmented multiple times will likely produce different augmented samples. This is a desirable property as producing the same synthetic sample over and over again would produce a lot of repetition in highly undersampled classes, and eventually lead to problems like overfitting as in each epoch the network would minimize loss for the same repeated sample, rather than learning from variations of that sample.

Considering the results from Figure 4.25, overall data augmentation does improve the performance in comparison with group 0, which does not employ any type of augmentation. Moreover, it seems the simpler augmentation approaches like group 1 and 2 perform the best, while more complex augmentations groups such as group 3 and 4 perform worst.

This results indicate that more conventional augmentation techniques alone perform better, which does make sense in the context of skin lesion classidicatino. As described in chapter 2 a common method used by dermathologists to identify skin lesions is the ABCDE method, which relies in features like the shape and color of the lesion to be diagnosed. Therefore, by changing such features we are potentially removing or changing important information about the lesion itself, which consequently will make the network perform worst. For instance, group 4 introduces noise inducing transformations, however, one cannot predict if the random erasing will erase important information from the lesion itself or if the distortion performed will alter the shape of the lesion in such a way that it is no longer characteristic of it's category. Similarly, by changing the colors of the lesion in group 2 we are pottentially changing important information about the colour of the lesion which is one of the most important features to consider when diagnosing a lesion **?**.

Furthermore, offline data augmentation is not the only method of applying data augmentation, specifically, online data augmentation is from the literature presented in chapter 2 a commonly used technique to reduce overfitting. In fact, as shown in Figure 4.26, online data augmentation significantly reduces the gap between train and validation both when employed with offline data augmentation or not, while also significantly increasing test performance. It is important to note that whenever data augmentation is applied in this experiment data augmentation group 1 is being used.

So far, results show that classification towards underepresented classes like dermafibroma has overall worse performance than classes like melanocytic nevus (see the confusion matrix from 4.20), which means that there is not enough samples to train a model for it to perform

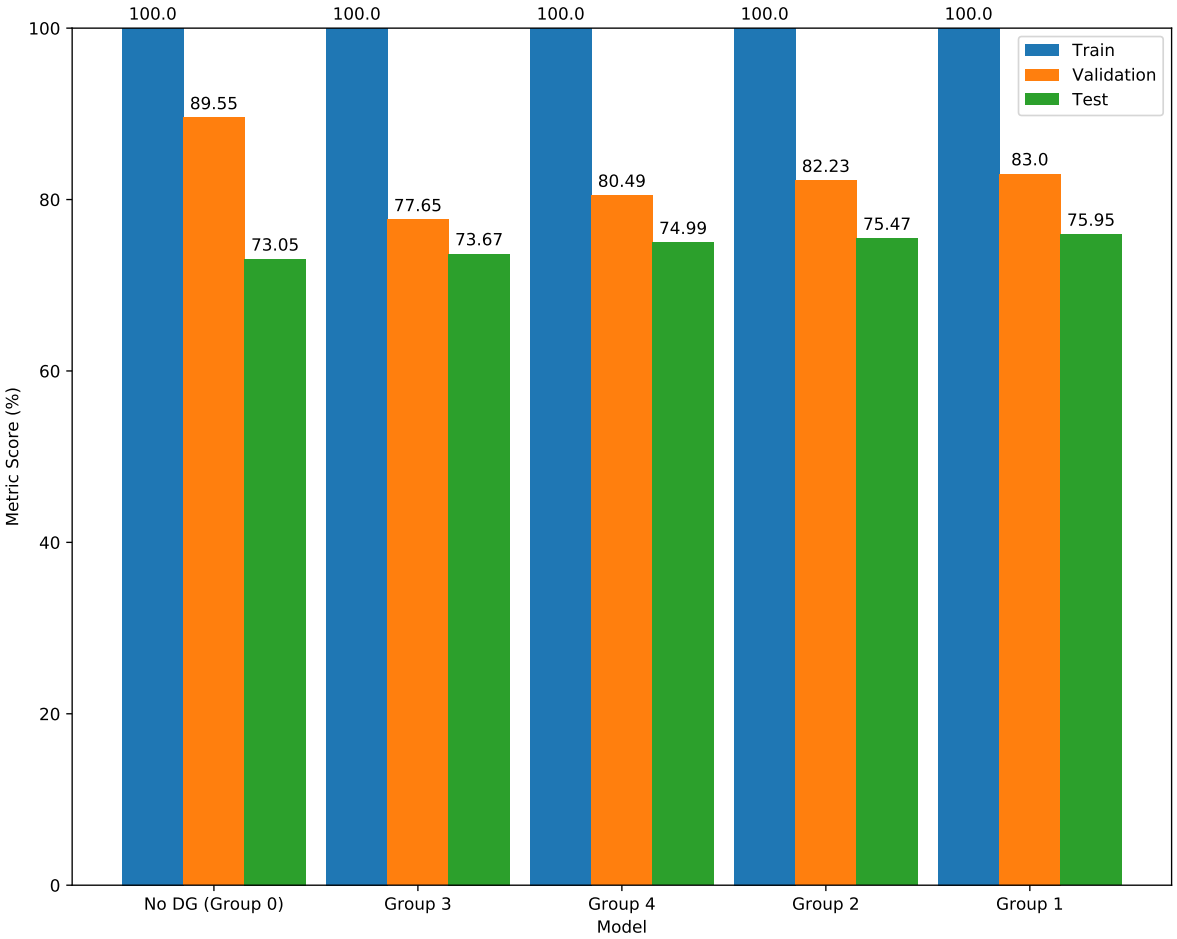


**Figure 4.25:** Balanced multi-class accuracy of train, validation and test sets of different offline data augmentation groups with a hyperparameter tuned DenseNet201 trained on 20264 balanced samples

well on these classes. Data augmentation, can alleviate this issue by generating interesting variations of samples that are part of underrepresented classes. For instance, Figure 4.27 shows an experiment where a balanced model with 2533 samples per class is compared against a unbalanced model of 20264 model by their balanced multi-class accuracy over the train, validation and test datasets. Results show that the balanced model will converge to a better local minimum in respect with all three sets.

However, by doing the same comparison but with the accuracy metric, results show that the accuracy over the unbalanced dataset are better than those from the balanced dataset (see Figure 4.28). It is evident from the results that offline data augmentation is improving performance on underrepresented classes shown by the $BMA$ increase, but by undersampling overrepresented classes, the accuracy takes a hit because these classes get slightly lower performance, which consequently does make a big impact on the accuracy metric as this metric does not average the results per class.

From the results shown, one can hypothesize that reducing undersampling from over-represented classes, while keeping classes balanced using offline data augmentation would yield better overall accuracy and slightly better $BMA$. Figure 4.28 shows the influence of

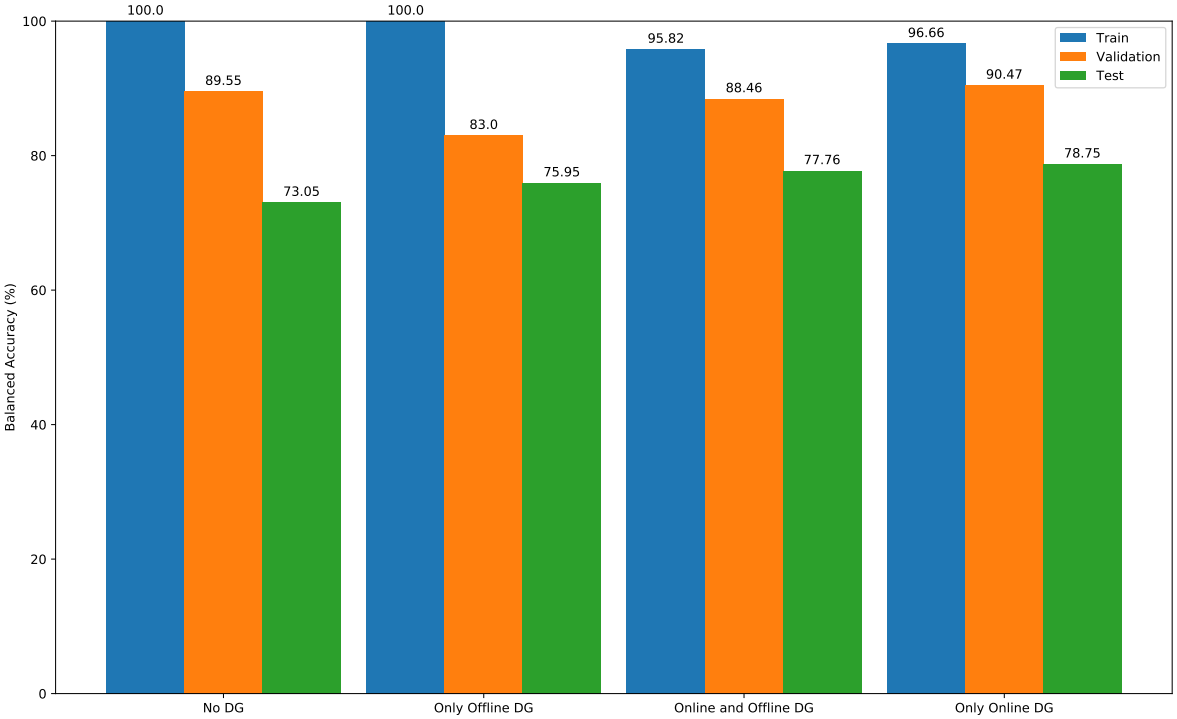


**Figure 4.26:** Balanced multi-class accuracy of train, validation and test sets of different combinations of offline and online data augmentation modes with a hyperparameter tuned DenseNet201 trained on 20264 balanced samples. No data augmentation is equivalent to augmentation group 0,

reducing undersampling on $BMA$ with the rightmost point (82400 samples) represents no undersampling being applied because each class has 10300 samples and the most prevalent class in the training dataset has 10300 samples (melanocytic nevus).

Considering the results from figure 4.29 it seems that increasing the dataset through augmentation does improve BMA on the test set to a certain set size, specifically around 40000 samples. Past that point, while the accuracy keeps improving the BMA does not change significantly. These results show that past 40000 samples the improvements come from the overepresented classes that were undersampled and are now getting better performance significantly improving accuracy. It is also important to notice how the BMA metric aproximates the accuracy metric when the datasets are balanced, which is the case of the validation and train sets, but in the test set they diverge as the test set is not balanced. Another important consideration is that between 20000 and 30000 samples, there is breakeven point between validation and train BMA and accuracy as the validation set starts attaining better performance than the train set. That breakeven point happens when the number of synthetic samples is superior to the number of original samples on the train and validation sets

Finally, the results on the test set show the remarkable impact of data augmentation used as a class balance measure, with a accuracy of $86,14\%$ and a balanced multi-class accuracy of $80,9\%$, which is a quite significant improvement from the fine-tuned model trained at section 4.2. Presumably, this jump in performance is due to an increase in sample size for

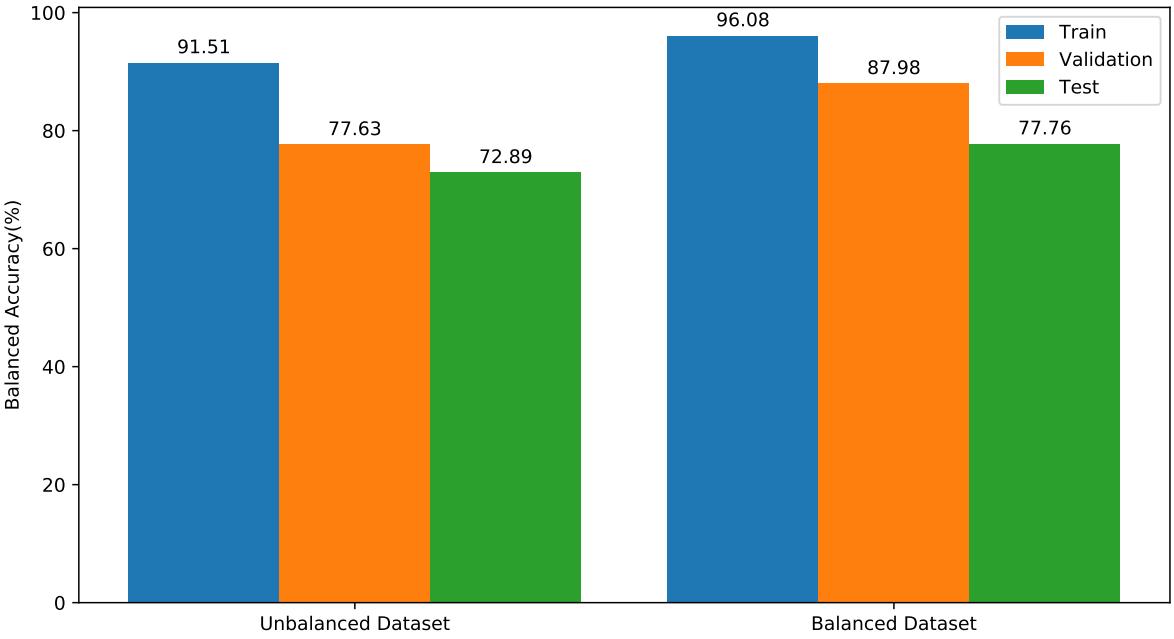


**Figure 4.27:** Comparison of the balanced multi-class accuracy of balanced and unbalanced datasets with 20264 samples for the train, validation and test sets.

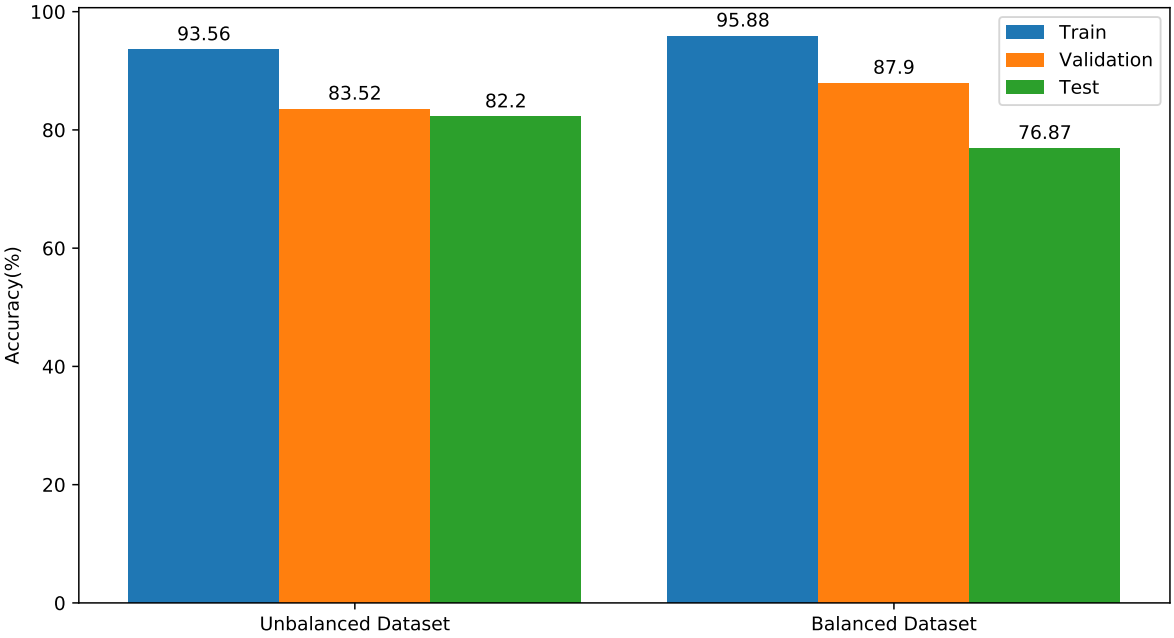


**Figure 4.28:** Comparison of the accuracy of balanced and unbalanced datasets with 20264 samples for the train, validation and test sets.

classes that are underrepresented on the original dataset. We can confirm such hypothesis by looking at Figure 4.30, which in comparison with Figure 4.20 shows that the network is much more capable of classifying correctly underrepresented classes.

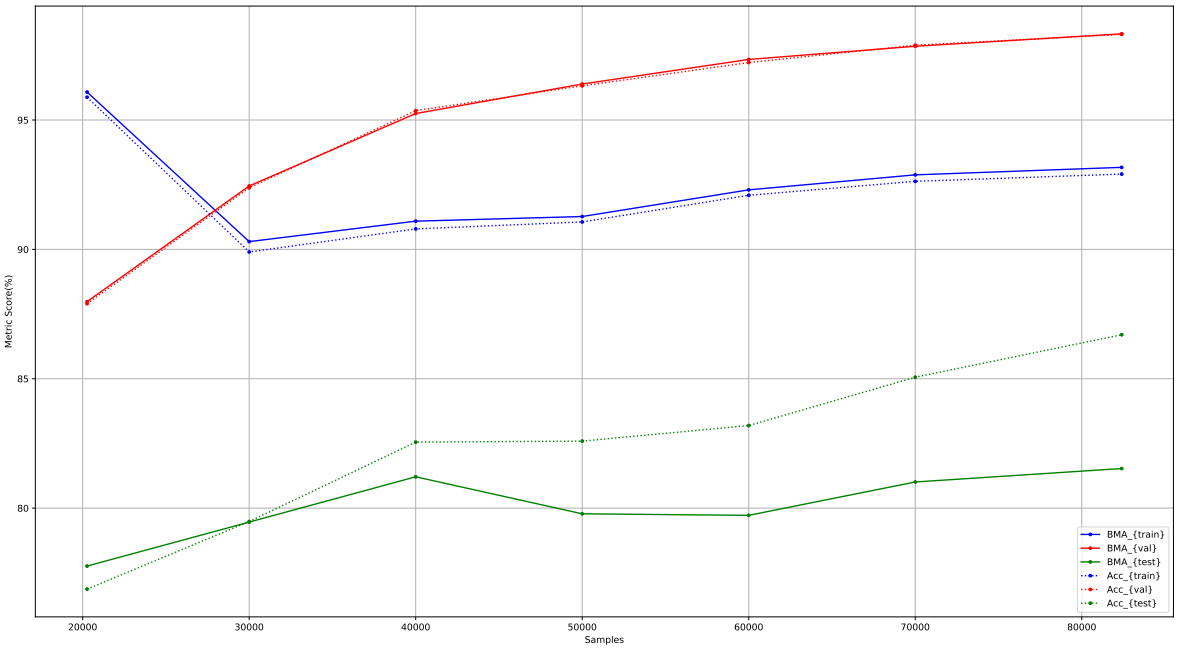


**Figure 4.29:** Comparison of the balanced multi-class accuracy for different balanced dataset sizes using the described oversampling and undersampling techniques

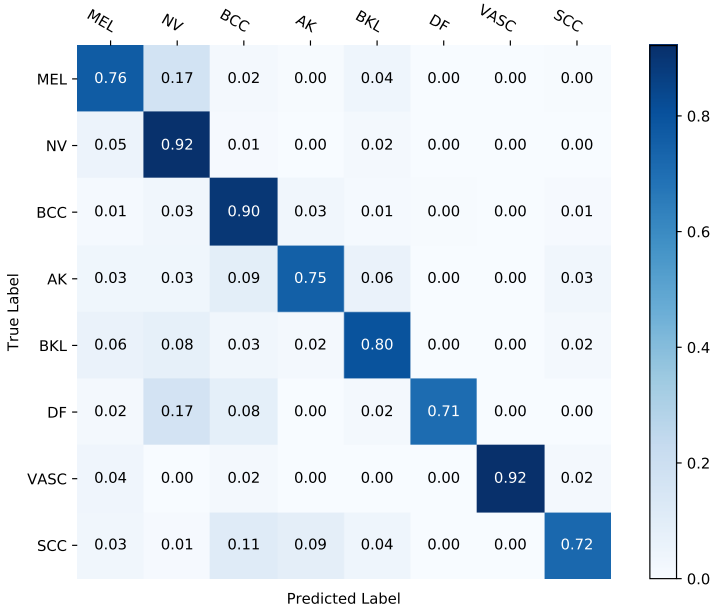


**Figure 4.30:** Confusion matrix of hyperparameter tuned DenseNet201 trained with 82400 balanced samples

## 4.4 Model ensemble

The literature suggests that the most successful approaches towards ISIC challenges are those based on ensemble of classifiers [41]. Often these classifiers have model and hyperparameter variations between each other, and are combined through either a weighted or unweighted softmax probability averaging scheme. Other approaches like voting schemes can also be used but those do not take into account the relative certainty of the model predictions. On that

account, based on Figure 4.3, several models

One important aspect to consider is how to produce meaningful variations between each model. By simply picking variations within a model architecture not enough variation is produced because often smaller models are just scaled versions of a baseline model, which means that averaging such models would not significantly increase the overall performance. Such is the case of models like VGG16 and VGG19 or EfficienNetB2 and EfficientNetB0. As such, to filter combinations of models the best performing model from each architecture in figure 4.3 was chosen, specifically, VGG16, ResNet152, DenseNet201, InceptionResNetV2 and EfficientNetB2. Additionally, for simplification all models will use the hyperparameters optimized for DenseNet201 in section section 4.2, while also using the best data augmentation schema from section 4.3. Results are shown in table 4.1.

| Model | Model $BMA$ | Model Accuracy |
|---|---|---|
| VGG16 | ≈0.7533 | ≈0.824 |
| ResNet152 | ≈0.797 | ≈0.858 |
| DenseNet201 | ≈0.815 | ≈0.867 |
| InceptionResNetV2 | ≈0.811 | ≈0.862 |
| EfficientNetB2 | ≈0.820 | ≈0.851 |
| Ensemble of best 2 | ≈0.842. | ≈0.878 |
| Ensemble of best 3 | ≈0.846 | ≈0.891 |
| Ensemble of best 4 | ≈0.841 | ≈0.894 |
| Ensemble of 5 | ≈0.836 | ≈0.893 |

**Table 4.1:** Ensemble comparison

Results from Table 4.1 show significant improvements of all ensembles over single model performance on both balanced multi-class accuracy and accuracy, with the ensemble of 3 being the best performing one reaching near 90% accuracy. One can observe that more models are not necessarily better, as the ensemble of the best 4 and the ensemble of 5 yield worse performance than the ensemble of the best 3. This can be attributed to the discrepancy of on the performance of VGG16 and ResNet152 in relation to the rest, which as a consequence of the averaging scheme bring the performance metrics down. However, the models of the ensemble of the best 3 all have similar performance individually, and overall benefit from being ensembled together, which can be attributed to the variation within each model architecture.

Finally, the confusion matrix of the ensemble of the best 3 models is shown in Figure 4.31. Results show that overall every class has a slightly better performance when compared to the single model performance of DenseNet201 at Figure 4.30.

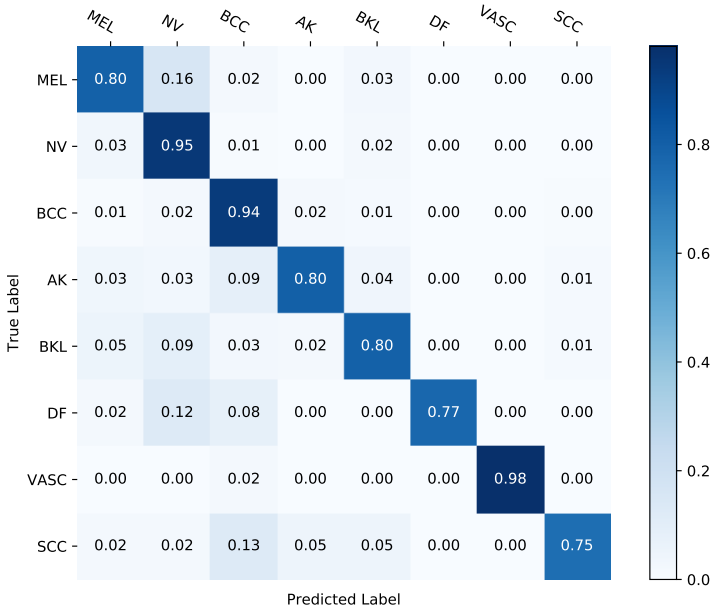


**Figure 4.31:** Confusion matrix of the ensemble of the best 3 pre-trained models (DenseNet201, EfficientNetB2 and InceptionResNetV2). Each model is trained with 82400 balanced samples with the best hyperparameters obtained from section 4.2

## 4.5 Out of distribution detection

The presented models shown that neural nets can be effective predictors to diagnose skin lesions. However, these models remain ignorant to its prediction confidence, (i.e. determining if a sample is or is not part of the training distribution). Although several approaches for dealing with out of distribution samples have been proposed on chapter 2, we will only explore 3 of them. Namely:

- Following Gessert et al's (ISIC 2019 winner) approach, we will train models with an additional outlier class, with additional samples.
- Following Zhou et al's submission, we are going to apply a top-1 softmax probability threshold.
- Finally, we are going to integrate the ODIN classifier [50], following Wang's approach [49].

### 4.5.1 Outlier class

### 4.5.2 Softmax threshold

### 4.5.3 ODIN classifier

CHAPTER 5

# Conclusion

In this section we present conclusions, final remarks, and point to directions for future work.

## 5.1 Discussion

In conclusion, several remarks are in order:

- conclusions

## 5.2 Future Work

A few lines of future work are possible:

- The performance of the models from ISIC 2019 might not represent real world scenarios. For instance, ISIC Archive is mostly composed of samples from a very narrow skin pigmentation interval, which might be impactful on diagnosis of skin lesions with other skin tones. As such, it would be interesting to analyze the performance of these models on a test set provided by an hospital or clinic.
- Two properties of skin lesion classification were left aside for our approach, namely the interpretability and the explainability of the models. These deep learning models can be considered a black box that does not provide an interpretation for it's diagnosis. As such, a desired feature of such networks would be for them to provide arguments for their decisions through the incorporation of medical knowledge.
- Although challenges like ISIC are great for pushing the performance of skin lesion classifiers forward, an important aspect to consider is deploying these systems into real world scenarios. Integrating these models into a tool with a user friendly interface is ultimately a priority, as it would benefit both dermatologists and patients by helping in the decision making process.
- unknown class...

# References


[1] *Skin Cancer Facts & Statistics - The Skin Cancer Foundation*, 2019. [Online]. Available: https://skincancer.org/skin-cancer-information/skin-cancer-facts/ (visited on 11/10/2019).

[2] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre, and A. Jemal, "Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries", *CA: A Cancer Journal for Clinicians*, vol. 68, no. 6, pp. 394–424, Nov. 2018, ISSN: 1542-4863. DOI: 10.3322/caac.21492.

[3] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks", *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017, ISSN: 0028-0836. DOI: 10.1038/nature21056. [Online]. Available: http://www.nature.com/articles/nature21056.

[4] H. A. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. Ben Hadj Hassen, L. Thomas, A. Enk, and L. Uhlmann, *Man against Machine: Diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists*, 2018. DOI: 10.1093/annonc/mdy166.

[5] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks For Large-Scale Image Recogntinion*, 2015. arXiv: 1409.1556v6. [Online]. Available: https://arxiv.org/pdf/1409.1556.pdf.

[6] M. Nielsen, *Neural Networks and Deep Learning*. 2018, pp. 389–411. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, Dec. 2015. DOI: 10.1007/s11263-015-0816-y. arXiv: 1409.0575.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2012, ISSN: 15577317. DOI: 10.1145/3065386.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, IEEE Computer Society, Dec. 2016, pp. 770–778, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[10] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks", Aug. 2016. arXiv: 1608.06993. [Online]. Available: http://arxiv.org/abs/1608.06993.

[11] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", May 2019. arXiv: 1905.11946. [Online]. Available: http://arxiv.org/abs/1905.11946.

[12] T. G. Dipanjan Sarkar, Raghav Bali, *Hands-On Transfer Learning with Python*, First. Packt Publishing, 2018.

[13] P. Ly, D. Bein, and A. Verma, *New Compact Deep Learning Model for Skin Cancer Recognition*, Aug. 2019. DOI: 10.1109/uemcon.2018.8796628.

[14] P. Marcelino, *Transfer learning from pre-trained models - Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 (visited on 12/02/2019).

[15] J. Grus, *Data Science From Scratch*. O'Reilly Media, 2015, ISBN: 9781491901427.

[16] A. Ng, *Feature selection, L 1 vs. L 2 regularization, and rotational invariance*.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", Jul. 2012. arXiv: 1207.0580. [Online]. Available: http://arxiv.org/abs/1207.0580.

[18] J. Brownlee, *Ensemble Learning Methods for Deep Learning Neural Networks*, 2018. [Online]. Available: https://machinelearningmastery.com/ensemble-methods-for-deep-learning-neural-networks/ (visited on 03/16/2020).

[19] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free", in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, Apr. 2019. arXiv: 1704.00109.

[20] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts", *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, Aug. 2016. arXiv: 1608.03983. [Online]. Available: http://arxiv.org/abs/1608.03983.

[21] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Yu Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, C. Yuen Koo, L. Lew, C. Mewald, A. Naresh Modi, N. Polyzotis, S. Ramesh, S. Roy, S. Euijong Whang, M. Wicke, J. Wilkiewicz, X. Zhang, and M. Zinkevich, "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform", 2017. DOI: 10.1145/3097983.3098021. [Online]. Available: http://dx.doi.org/10.1145/3097983.3098021.

[22] F. Chollet *et al.*, *Keras*, \url{https://github.com/fchollet/keras}, 2015.

[23] F. Bastien, N. Pascal Lamblin, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, *Theano: new features and speed improvements*. arXiv: 1211.5590v1.

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. D. Facebook, A. I. Research, Z. Lin, A. Desmaison, L. Antiga, O. Srl, and A. Lerer, "Automatic differentiation in PyTorch", Tech. Rep., 2017.

[25] J. Jaworek-Korjakowska and P. Kleczek, "ESkin: Study on the smartphone application for early detection of malignant melanoma", *Wireless Communications and Mobile Computing*, vol. 2018, 2018, ISSN: 15308677. DOI: 10.1155/2018/5767360.

[26] *DermEngine | Visual Search*. [Online]. Available: https://www.dermengine.com/en-ca/visual-search (visited on 11/21/2019).

[27] C. Barata, M. Ruela, M. Francisco, T. Mendonca, and J. S. Marques, "Two systems for the detection of melanomas in dermoscopy images using texture and color features", *IEEE Systems Journal*, vol. 8, no. 3, pp. 965–979, 2014, ISSN: 19379234. DOI: 10.1109/JSYST.2013.2271540.

[28] S. Pathan, K. G. Prabhu, and P. C. Siddalingaswamy, *Techniques and algorithms for computer aided diagnosis of pigmented skin lesions—A review*, Jan. 2018. DOI: 10.1016/j.bspc.2017.07.010.

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

[30] H. Greenspan, B. Van Ginneken, and R. M. Summers, "Guest Editorial Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique", *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, May 2016, ISSN: 1558254X. DOI: 10.1109/TMI.2016.2553401.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database", Institute of Electrical and Electronics Engineers (IEEE), Mar. 2010, pp. 248–255. DOI: 10.1109/cvpr.2009.5206848.

[32] T. J. Brinker, A. Hekler, J. S. Utikal, N. Grabe, D. Schadendorf, J. Klode, C. Berking, T. Steeb, A. H. Enk, and C. von Kalle, "Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review.", *Journal of medical Internet research*, vol. 20, no. 10, e11936, Oct. 2018, ISSN: 1438-8871. DOI: 10.2196/11936. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/30333097.

[33] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions", *Scientific Data*, vol. 5, Aug. 2018, ISSN: 20524463. DOI: 10.1038/sdata.2018.161. arXiv: 1803.10417.

[34] *ISIC 2019*, 2019. [Online]. Available: https://challenge2019.isic-archive.com/ (visited on 11/23/2019).

[35] *Task 3: Lesion Diagnosis | ISIC 2018*, 2018. [Online]. Available: https://challenge2018.isic-archive.com/task3/ (visited on 11/23/2019).

[36] A. Nozdryn-Plotnicki, J. Yap, and W. Yolland, *Ensembling Convolutional Neural Networks for Skin Cancer Classification*, 2018.

[37] E. Finlayson, Graham and Trezzi, "Shades of Gray and Colour Constancy", *Proceedings of the 12th Color Imaging Conference*, pp. 37–41, 2004.

[38] N. Gessert, T. Sentker, F. Madesta, R. Schmitz, H. Kniep, I. Baltruschat, R. Werner, and A. Schlaefer, "Skin Lesion Diagnosis using Ensembles, Unscaled Multi-Crop Evaluation and Loss Weighting", Aug. 2018. arXiv: 1808.01694. [Online]. Available: http://arxiv.org/abs/1808.01694.

[39] M. A. A. Milton, "Automated Skin Lesion Classification Using Ensemble of Deep Neural Networks in ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection Challenge", Jan. 2019. arXiv: 1901.10802. [Online]. Available: http://arxiv.org/abs/1901.10802.

[40] A. Bissoto, F. Perez, V. Ribeiro, M. Fornaciali, S. Avila, and E. Valle, "Deep-Learning Ensembles for Skin-Lesion Segmentation, Analysis, Classification: RECOD Titans at ISIC Challenge 2018", Aug. 2018. arXiv: 1808.08480. [Online]. Available: http://arxiv.org/abs/1808.08480.

[41] P. Tschandl, N. Codella, B. N. Akay, G. Argenziano, R. P. Braun, H. Cabo, D. Gutman, A. Halpern, B. Helba, R. Hofmann-Wellenhof, A. Lallas, J. Lapins, C. Longo, J. Malvehy, M. A. Marchetti, A. Marghoob, S. Menzies, A. Oakley, J. Paoli, S. Puig, C. Rinner, C. Rosendahl, A. Scope, C. Sinz, H. P. Soyer, L. Thomas, I. Zalaudek, and H. Kittler, "Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: an open, web-based, international, diagnostic study", *The Lancet Oncology*, vol. 20, no. 7, pp. 938–947, Jul. 2019, ISSN: 14745488. DOI: 10.1016/S1470-2045(19)30333-X.

[42] S. S. Han, M. S. Kim, W. Lim, G. H. Park, I. Park, and S. E. Chang, "Classification of the Clinical Images for Benign and Malignant Cutaneous Tumors Using a Deep Learning Algorithm", *Journal of Investigative Dermatology*, vol. 138, no. 7, pp. 1529–1538, Jul. 2018, ISSN: 15231747. DOI: 10.1016/j.jid.2018.01.028.

[43] N. Gessert, M. Nielsen, M. Shaikh, R. Werner, and A. Schlaefer, *Skin Lesion Classification Using Loss Balancing and Ensembles of Multi-Resolution EfficientNets*, 2019.

[44] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout", Aug. 2017. arXiv: 1708.04552. [Online]. Available: http://arxiv.org/abs/1708.04552.

[45] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization", in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR, Dec. 2015. arXiv: 1412.6980.

[46] S. Zhou, Y. Zhuang, and R. Meng, *Multi-Category Skin Lesion Diagnosis Using Dermoscopy Images and Deep CNN Ensembles*, 2019.

[47] F. Pollastri, J. Maroñas, M. Parreño, F. Bolelli, R. Paredes, C. Grana, and A. Albiol, "AImageLab-PRHLT at ISIC Challenge 2019", Tech. Rep., 2019.

[48] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke, "Out-of-Distribution Detection Using an Ensemble of Self Supervised Leave-out Classifiers", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11212 LNCS, pp. 560–574, Sep. 2018. arXiv: 1809.03576. [Online]. Available: http://arxiv.org/abs/1809.03576.

[49] H.-W. Wang, "Skin Lesion Classification using Ensemble of Convolutional Neural Networks and Out-of-Distribution Detector", Tech. Rep., 2019.

[50] S. Liang, Y. Li, and R. Srikant, "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks", *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, Jun. 2017. arXiv: 1706.02690. [Online]. Available: http://arxiv.org/abs/1706.02690%20https://github.com/facebookresearch/odin.

[51] M. Combalia, N. C. F. Codella, V. Rotemberg, B. Helba, V. Vilaplana, O. Reiter, C. Carrera, A. Barreiro, A. C. Halpern, S. Puig, and J. Malvehy, *BCN20000: Dermoscopic Lesions in the Wild*, Aug. 2019. arXiv: 1908.02288. [Online]. Available: http://arxiv.org/abs/1908.02288.

[52] N. C. F. Codella, *Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)*, Oct. 2017. arXiv: 1710.05006. [Online]. Available: http://arxiv.org/abs/1710.05006.

[53] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: BEYOND EMPIRICAL RISK MINIMIZATION", Tech. Rep. arXiv: 1710.09412v2. [Online]. Available: https://github.com/facebookresearch/mixup-cifar10..

[54] F. Maia, "A Study of Transfer Learning for Skin Lesion Classificatio", PhD thesis, University of Aveiro, 2019.

[55] G. C. Cawley and N. L. C. Talbot, "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation", *Journal of Machine Learning Research*, vol. 11, pp. 2079–2107, 2010.