Universidade de Aveiro
2020

Departamento de Eletrónica,
Telecomunicações e Informática

Fábio
Santos

Aprendizagem profunda para o diagnóstico de múltiplas classes de lesões de pele

Deep learning for multi-class skin lesion diagnosis

Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2020

Fábio Santos

Aprendizagem profunda para o diagnóstico de múltiplas classes de lesões de pele

Deep learning for multi-class skin lesion diagnosis

"*Our intelligence is what makes us human, and AI is an extension of that quality*"

— Yann LeCun

Universidade de Aveiro
2020

Departamento de Eletrónica,
Telecomunicações e Informática

Fábio Santos

Aprendizagem profunda para o diagnóstico de múltiplas classes de lesões de pele

Deep learning for multi-class skin lesion diagnosis

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 Filipe Miguel Teixeira Pereira da Silva, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Doutora Pétia Georgieva, Professora auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais.

o júri / the jury

presidente / president

Prof. Doutor Luís Filipe de Seabra Lopes
Professor Associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor João Paulo Morais Ferreira
Professor Adjunto do Instituto Superior de Engenharia de Coimbra

Prof. Doutor Filipe Miguel Teixeira Pereira da Silva
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos / acknowledgements

Quero deixar um grande agradecimento ao orientador Filipe Silva e à co-orientadora Pétia Georgieva por toda a dedicação e apoio prestado no desenvolvimento desta dissertação. Quero também agradecer ao Professor Vitor Santos (DEM) e à equipa do LAR do departamento de Engenharia Mecânica, pela disponibilidade dos recursos computacionais necessários para a conclusão desta dissertação. Por fim, quero deixar um especial agradecimento aos meus amigos e família que desde cedo me deram o suporte necessário para que seja bem sucedido nos trabalhos que realizo.

Palavras Chave

classificação de lesões de pele, classificação de múltiplas classes, aprendizagem profunda, rede neuronal convolucional, aprendizagem por transferência, desempenho de generalização.

Resumo

Aprendizagem profunda é um ramo de aprendizagem automática que tem crescido rapidamente e que tem sido usada em problemas relacionados com imagem médica, tais como a deteção de cancro da pele. Durante muito tempo, o diagnóstico de lesões de pele através de imagens clínicas usando aprendizagem profunda era algo considerado inalcançável. No entanto, avanços recentes em redes neuronais profundas permitiram obter resultados de referência em tarefas de classificação. Além disto, existe uma necessidade crescente de sistemas capazes de automatizar este tipo de diagnósticos para reduzir a taxa de mortalidade causada por cancro da pele. Isto proporcionará suporte a dermatologistas no processo de decisão e a pacientes que não tenham fácil acesso a médicos especialistas. Esta dissertação apresenta uma abordagem sistemática tendo em vista o desenvolvimento de um classificador multiclasse (8 classes) para o diagnóstico automático de lesões de pele. O estudo visa analisar um conjunto de metodologias de aprendizagem profunda que permitam responder a questões relacionadas com a sua eficácia. Os resultados indicam que os modelos pre-treinados, baseados em arquiteturas convolucionais, têm um impacto significativo no desempenho desde que sejam devidamente reaproveitados usando aprendizagem por transferência. Este trabalho proporciona uma visão mais clara sobre a influência de diferentes métodos na capacidade de generalização destes modelos. Neste contexto, o impacto de diferentes métodos de processamento de imagem para o aumento de dados é revelado. Por um lado, o balanceamento de classes através do aumento de dados offline permite melhorar significativamente o desempenho de classes sub-representadas. Por outro lado, o aumento de dados on-line traz melhorias consideráveis na redução do sobreajuste (*overfitting*). Adicionalmente, é fornecida uma análise prática de métodos de *ensemble* no contexto do diagnóstico de lesões de pele. Os resultados obtidos com um modelo pre-treinado e com o *ensemble* superam o atual estado da arte da competição ISIC 2019, com uma acurácia balanceada (*balanced accuracy*) de 0,815 e 0,846, respetivamente. Finalmente, foram feitas experiências com diferentes algoritmos para detetar dados que não pertencem à distribuição que resulta das 8 classes originais. Os resultados obtidos contribuem para a compreensão da eficácia destes métodos no contexto de classificadores de lesões de pele.

Keywords skin lesion diagnosis, multi-class classification, deep learning, convolutional neural network, transfer learning, generalization performance.

Abstract

Machine learning, more specifically, deep learning is a fast-growing field that is being used for multiple medical imaging related problems, such as the early detection of skin cancer. For a long time, automated diagnosis of skin lesions from clinical images through deep learning was considered to be out of reach. However, recent advancements in deep neural networks allowed it to achieve state-of-the-art results on classification challenges and they have the potential to change the landscape of dermatology care. Moreover, there is a growing need for classification systems to reduce fatality rates of skin cancer by providing support for both dermatologists in the decision-making process and for patients that do not have access to expert physicians. This dissertation presents a systematic approach towards an multi-class (8 classes) deep learning classifier of skin lesions for the ISIC 2019 benchmark challenge. It attempts to study major open research points related to the effectiveness of state-of-the-art deep learning methods. The results indicate that recent CNN architectures can have a significant impact on the overall performance of deep learning based classifiers. However, these pre-trained models should be carefully re-purposed towards skin lesion classification through transfer learning methods. This work provides insight into major ways of improving the generalization performance of deep learning models towards skin lesion diagnosis. In this context, the impact of different image processing augmentation methods for offline and online data augmentation is studied. On the one hand, class balancing through offline data augmentation can significantly improve the generalization performance of underrepresented classes. On the other hand, online data augmentation brings considerable improvements towards overfitting reduction. Furthermore, this work provides a practical analysis of ensembles in the context of skin lesion diagnosis. Both the single model and the ensemble-model approach outperform the current state-of-the-art for the ISIC 2019 challenge, with a balanced multi-class accuracy of 0.815 and 0.846, respectively. Finally, experiments were made with different ways to detect out of training distribution data. The results contribute towards the understanding of the effectiveness of out-of-distribution detection methods in the context of deep learning based skin lesion classifiers.

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
GAN	Generative Adversarial Networks
SVM	Support Vector Machine
CPU	Central Processing Unit
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
RAM	Random Access Memory
FLOPS	FLoating point Operations Per Second
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
PPV	Positive Predictive Value
NPV	Negative Predictive Value
AUC	Area Under Curve
ROC	Receiver Operating Characteristic
BMA	Balanced Multi-class Accuracy
LAR	Laboratory for Automation and Robotics
ISIC	International Skin Imaging Collaboration
ILSVRC	ImageNet Large Scale Visual Recognition Competition
MNIST	Modified National Institute of Standards and Technology
HAM10000	Human Against Machine with 10000 images
ODIN	Out-of-DIstribution detector for Neural networks
TDS	Total Dermoscopic Score
SCC	Squamous Cell Carcinoma
BCC	Basal Cell Carcinoma

CHAPTER 1

Introduction

In this chapter some background will be given for the problem statement, as well as the motivation and objectives behind this work.

1.1 Background

Skin cancer is the out-of-control growth of abnormal cells in the outermost skin layer, caused by mutations triggered within the genetic code [1]. These mutations lead the skin cells to multiply rapidly and form malignant tumors. It is currently the most common type of cancer [2] with melanoma being the most deadly form that accounts for about 75% of skin cancer deaths, even though it only represents 5% of all skin cancer cases [3]. Other forms of skin cancer such as the Basal Cell Carcinoma (BCC) or the Squamous Cell Carcinoma (SCC) are not as deadly but are much more common. BCCs are the most common type of skin cancer and have the potential to disfigure the skin and become dangerous [2]. Similarly, SCCs are the second most common form of this decease and, if left untreated, can destroy nearby healthy tissue, spread to other parts of the body, and eventually lead to death [1].

A recent study observed that in the United States of America alone, there are 5.4 Million new cases of skin cancer every year [2]. However, a bigger problem is that the incidence rates of skin cancer keep rising with currently 1 in 5 persons developing skin cancer until the age of 70 [1]. In Europe, melanoma incidence rates also manifest heavily, as each year 100000 people are diagnosed with melanoma, and approximately 22000 people die annually from this form of skin cancer [4].

Even though skin cancer can be deadly in late stages, if detected early there is a high chance of survival. For example, there is a 23% chance of surviving a melanoma case if detected in the late stages, but if detected early, the five-year survival rate (*i.e.*, percentage of people alive after five years) is approximately 98% [1]. Therefore, the early detection of skin cancer is a top priority in order to increase the overall survival rate of the population.

Skin cancer can be detected by dermatology professionals by performing a visual examination of skin lesions. However, some authors believe that the dermatologist's experience

directly impacts his diagnostic accuracy [5], which implies that different physicians might make a different diagnosis for the same lesion. Some works have corroborated this in practice. For example, Argenziano *et al.* in their study noted that dermatologists have a 65% to 80% interval of accuracy in melanoma diagnosis [6]. However, this rate can be overall increased with the supplement of dermatoscopic images [7]. These type of images are taken with a special high-resolution and magnifying camera in a controlled light environment, where reflections on the skin are minimized which causes deeper skin layers to be visible.

Automated diagnosis of skin lesions through the dermatoscopic and non-dermatoscopic images has been achieving significant progress over the years, as demonstrated by some state-of-the-art surveys [8][9][10]. Typically, computer vision algorithms are used to analyze images and extract structural information [11], such as the ABCDE (Asymmetry, Border, Color, Dermoscopic structure, and Evolving) rule [12] or the CASH (color architecture, symmetry, and homogeneity) algorithm. These algorithms are then integrated within systems to provide quantification of features for physician assessment and for provisional diagnosis.

However, recent studies show that more complex structures called Convolutional Neural Networks (CNNs) are being applied to classify skin lesions with remarkable performance in comparison to human experts [3][5][13]. Moreover, the interest on this topic is well demonstrated by the large number of papers related to deep learning published in the context of the ISIC challenges [14]. This paradigm shift became apparent in 2016, for the International Symposium on Biomedical Imaging (ISBI) benchmark challenge towards melanoma detection. From 25 teams, all of them employed CNNs, instead of hand-crafted computer vision algorithms or other machine learning methods [15].

CNNs are a type of deep learning architecture. In turn, deep learning refers to computational models composed of multiple processing layers capable of learning representations of data with multiple levels of abstraction [16]. Deep learning topologies have a considerable advantage over traditional Artificial Neural Networks (ANNs), namely, they do not require the selection of an appropriate set of features from which the algorithm learns. This task involves extracting knowledge or information which is implicit in the raw data, requiring careful engineering and knowledge of the problem domain. Some developments allowed CNNs to emerge as an feasible approach to computer vision classification problems, such as the use of the Graphics Processing Unit (GPU) to speedup computations and the development of high-level software modules to train deep neural networks.

1.2 Motivation

Current implementations of deep learning models for skin lesion diagnosis still present major challenges to overcome. The biggest one is the requirement for large amounts of training data in order to achieve superior performance to other methods. However, publicly available datasets for medical imaging related problems are typically small compared to the needs of training a deep CNN from scratch [17]. Even though training a model from scratch is a feasible approach for big datasets (*e.g.*, ImageNet [18]), alternative approaches like transfer learning present an workaround for small datasets. Transfer learning is a technique in which

one can leverage the knowledge obtained from a deep model previously trained on a large labelled dataset towards a new classification task [19]. Furthermore, transfer learning does not require as much machine learning expertise as designing a model from scratch, tuning its hyperparameters, and making thorough discussions about the results. For this reason, in recent years, transfer learning based approaches have become prevalent for automated skin lesion diagnosis [3][5][20][21].

However, some questions remain open in the context of transfer learning based models for skin lesion classification, namely: "*Which strategy should one chose to train these transfer learning based models?*", "*What pre-trained models and architectures bring benefits to the classification of skin lesions?*", "*How does model depth affects the performance of transfer learning based deep learning models?*", and "*How can hyperparameter optimization help transfer learning based approaches achieve higher generalization performance?*".

In addition to transfer learning, a commonly used method to ease the requirement for large amounts of data is data augmentation [22]. This term involves a large range of methodologies which aim to create synthetic samples that allows a model to attain better generalization performance. Furthermore, skin lesion datasets are often highly unbalanced which can cause the model to optimize its results for overrepresented classes. Several approaches attempt to solve this issue by class balancing samples through data augmentation [23]. However, some questions remain open about its usage, namely: "*Can class balancing through data augmentation help increase generalization performance of underrepresented classes?*", "*What is the influence of dataset size on generalization performance?*", "*What is the influence of data augmentation in deep learning for small datasets?*", "*Which data augmentation image processing algorithms should one apply for the problem of skin lesion classification?*".

Moreover, the top approaches towards skin lesion classification systems of benchmark challenges such as the ISIC 2019 [24], usually employ model ensembling methods in order to attain better generalization performance [21] [25] [26]. However, one could ask the following open questions: "*How can one employ these type of techniques to get meaningful improvements on the generalization performance?*", "*Do the obtained improvements justify the disadvantages of using these types of methods?*", and "*Are these approaches practical to be deployed into the clinical workflow?*". Ultimately, the goal of all of these questions is to further improve generalization performance of deep learning models.

However, in a production environment, samples of skin lesions may be taken under far different conditions from their training dataset (*e.g.*, lighting conditions) or might not be part of the original training categories. Work in this field reports that deep learning models are highly sensitive to a different sample distribution from their original training distribution [14][27]. This is a known problem of deep learning models, which out-of-distribution detection methods attempt to solve. Ideally, these types of methods could flag out-of-distribution samples for further inspection of a dermatologist, which would ultimately augment his capabilities, rather than replace them. However, some questions remain open towards this methods in the context of skin lesion classification: "*What methods can one use to detect out of training distribution samples?*" and "*What is the effectiveness of these methods for skin lesion diagnosis?*".

1.3 Objectives

A strong assumption can be made that the recent advancements of deep learning based methods have the potential to change the landscape of skin lesion diagnosis because it would minimize the time it takes to detect skin cancer cases. However, one must first closely study the impact of the presented problems and the different strategies to deal with them, as it will likely lead to useful insights and contributions to the current state-of-the-art methods. Therefore, the following key objectives are highlighted for this work:

- Study the impact of transfer learning methodologies, pre-trained model architectures and parameters in the context of skin lesion classification;
- Study the impact of data augmentation, class balancing, and ensembling techniques as methods to improve performance for skin lesion classification;
- Experiment with different strategies to identify out of training distribution samples from the test set;
- Create a comprehensive approach for the ISIC 2019 benchmark challenge that provides some insight into ways of optimizing generalization performance.

1.4 Outline

Considering the presented objectives, the remainder of this dissertation is organized as follows:

- Chapter 2 introduces the reader to deep learning concepts and techniques relevant to the current state-of-the-art methods which automate skin lesion diagnosis using deep learning;
- Chapter 3 provides an overview of recent progress in the automated diagnosis of skin lesions;
- Chapter 4 describes the experimental scope of this work and the setup used for the experiments;
- Chapter 5 presents a range of experiments with different pre-trained models, transfer learning methods and hyperparameters;
- Chapter 6 studies the impact of dataset size, data augmentation methods, ensembling techniques and methodologies to detect out of training distribution samples;
- Chapter 7 offers final remarks, key takeaways, limitations of this research, and directions for future work.

CHAPTER 2

Methods and Materials

This chapter explains the necessary fundamentals of deep learning to familiarize the reader with the methods used throughout this work. More specifically, Section 2.1 provides an explanatory overlook of the fundamental concepts of feedforward neural networks. Section 2.2 touches on the fundamental concepts used by CNNs, as well as describe state-of-the-art pre-trained model architectures used throughout this work. The Section 2.3 explores the concept of repurposing pre-trained models from CNN architectures trained on generic datasets. Section 2.4 explores a common problem for machine learning algorithms in the context of deep learning, namely, the bias-variance tradeoff and ways to deal with such a problem. Afterward, Section 2.5 presents different ways to improve a model's performance through ensemble learning. Next, Section 2.6 introduces the reader to commonly used metrics to assess performance of a deep learning model. Finally, Section 2.7 describes techniques to deal with the detection of out of the training distribution samples, which will be one of the focus points of this work.

2.1 Artificial Neural Networks

Artificial Neural Networks compose a category of machine learning algorithms that are inspired by biological neural networks. These structures are comprised of multiple layers, each composed by multiple neurons that can be interpreted as a function described by parameters.

Within a neuron, each input x is multiplied by a learnable matrix of weights w, 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 (also called activation) according to a activation function g which introduces non-linearity. Mathematically speaking, the output of a neuron can be expressed as:

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

One can look at the process of training an Artificial Neural Network (ANN) as an iterative process that tries to optimize parameters in order to minimize a cost function. There are many different topologies of ANNs. One of the most common is the fully connected neural network, where each neuron is connected to every other neuron in the previous layer (see an example in Figure 2.1).

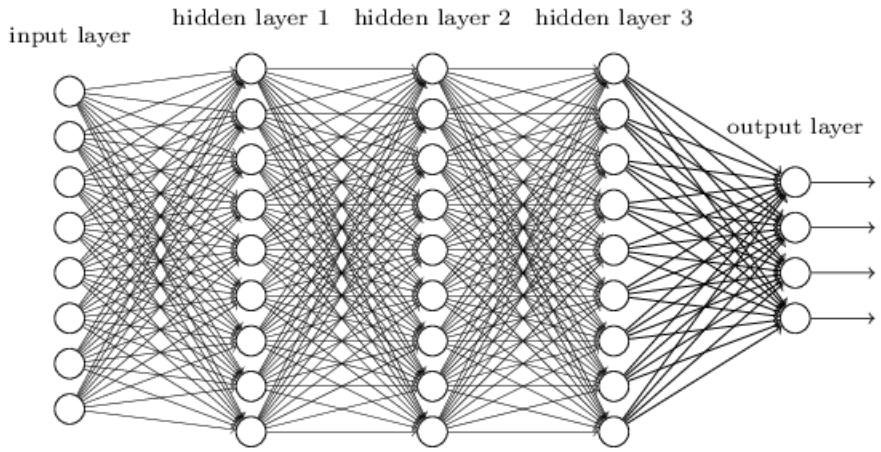

Figure 2.1: Example of a feedforward fully-connected artificial neural network taken from Michael Nielsen [28].

According to the universal approximation theorem, feedforward ANNs, a type of ANN where connections between the nodes do not form a cycle, can approximate any function with just one layer. However, for complex functions an increased hidden layer width may be required, which can ultimately be inefficient [29]. Therefore, networks with more hidden layers and with an organization that allows them to create levels of abstraction are often used to solve more complex problems. Such network topologies are called deep neural networks and are part of the broader field of deep learning. There are other deep learning topologies such as deep belief networks, recurrent neural networks, or convolutional neural networks.

2.1.1 Activation Functions

The activation function of a neuron defines the output (activation) of that neuron given an input $z = \sum_i^n x_i w_i + b$. In this context, several activation functions g have been proposed for ANNs, namely:

- The sigmoid function $g(z) = (\frac{1}{1+exp^{-z}})$. Provides smooth gradient between 0 and 1 values, but suffers from computation issues like the vanishing gradients problem;
- The tanh function $g(z) = (\frac{e^z - e^{-z}}{e^z + e^{-z}})$. Like the sigmoid except that output values range from -1 to 1, meaning its center is at 0;
- The ReLU function $g(z) = max(0, z)$. A network with such neurons is able to overcome numerical computation issues like the exploding and vanishing of gradients (typically associated with activation functions like the sigmoid function). Some authors also report that it allows faster convergence in comparison with other activation functions like tanh [30];

- The softmax function $g(z_i) = \frac{exp(z_i)}{\sum_i exp(z_j)}$. It is typically used in the output layer for neural networks which classifies 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 [16].

2.1.2 Optimization Algorithms

Optimization refers to the process of tuning the parameters of a network in order to improve a specific performance metric (*e.g.*, accuracy). The ideal approach of an optimization algorithm would be to find the global minimum of the cost function, but such an approach is not efficient because it would require testing every possible combination of parameters. Rather, an approximate combination of optimal parameters is often satisfiable, which in turn results in a local minimum of the cost function. Currently, stochastic gradient descent with and without momentum, RMSProp, and Adam belong to the most popular optimization strategies used for deep learning based models [16].

Gradient descent is an algorithm that updates the parameters of a function in the direction in which it decreases most rapidly. Let there be a function f to be minimized, and a vector of parameters w with n elements. One can mathematically look at gradient descent as:

$$w_{t+1} = w_t - \eta \nabla f(w_t) \tag{2.2}$$

where the expression $-\eta \nabla f(w_t)$ can be seen as the step in the parameter space to take at iteration t, η is the learning rate that controls the magnitude of the update, and the expression $\nabla f(w_t)$ is the vector of first partial derivatives:

$$\nabla f(w_t) = (\frac{\delta f(w_t)}{\delta f(w_{t,1})}, ..., \frac{\delta f(w_t)}{\delta f(w_{t,n})}) \tag{2.3}$$

However computing $\nabla f(w_t)$ requires calculating gradients over the whole data which is computationally expensive. Alternatively, in mini-batch gradient descent, the gradient is estimated over a mini-batch (a small number of randomly selected samples from the original dataset), which in theory approximates the true gradient. Mathematically speaking, the true gradient ∇f calculated over m samples can be approximated by simply calculating the gradient over a mini-batch of m' samples:

$$\nabla f \approx \frac{\sum_{i=1}^{m'} \nabla f_i}{m'} \approx \frac{\sum_{i=1}^{m} \nabla f_i}{m} \tag{2.4}$$

A variation of this method adds momentum to the movement of the gradient and is called stochastic gradient descent with momentum [31]. This method is analogous to a ball moving on a surface with multiple valleys, accelerating on steep slides and decelerating when it reaches a valley. The intuition behind this method is to add inertia to the gradient descent

so that it smooths the overall trajectory, in order to eventually find better convergence points. Mathematically, it can be described as:

$$\begin{aligned} v_0 &= 0 \\ v_{t+1} &= -\beta v_t + \eta \nabla f(w_t) \\ w_{t+1} &= w_t - v_{t+1} \end{aligned} \tag{2.5}$$

where the parameter η (learning rate) defines the impact of the current gradient and β defines the impact of the previous gradients.

Deep neural networks often have to solve optimization problems with large amounts of dimensions, which poses the question of whether or not it makes sense to use the same learning rate for each of those dimensions, as each one might have different sensitivities. Taking this into consideration, optimization algorithms such as AdaGrad [32] adjusts the learning rate of each parameter individually:

$$w_{t+1} = w_t - \eta G_t^{-\frac{1}{2}} g_t \tag{2.6}$$

where $g_t = \nabla f(w_t)$, $G_t = \sum_{i=1}^{t} g_i g_i^T$ and η is the global learning rate.

However, the Adagrad optimization algorithm instigates a decrease in the learning rate which is caused by the accumulation of all past gradients in the denominator. This means that at a certain point in the training procedure, the model becomes unable to learn. RMSProp solves this issue by letting the sum of the accumulated gradients decay (see expressions in 2.7). Therefore, the gradient accumulation is an exponentially weighted moving average of the squared gradients.

$$\begin{aligned} R[g^2]_t &= \rho R[g^2]_{t-1} + (1-\rho) g_t^2 \\ w_{t+1} &= w_t - \eta * \frac{g_t}{\sqrt{R[g^2]_t}} \end{aligned} \tag{2.7}$$

where $g_t = \nabla f(w_t)$, the $R[g^2]_t$ is the the running average at time step t (at $t = 0$, the $R[g^2]$ is 0), ρ denotes the weight given to the current iteration's gradient in relation to the past history and η is the global learning rate.

Finally, Adam [33] can be looked at as a combination of RMSprop and stochastic gradient descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using the moving average of the gradient. The expressions in 2.8 describe the weight updates, where S_t is the first-order moment (the mean) and R_t is the second-order moment (the uncentered variance) of the gradients.

$$\begin{aligned} S_t = \rho_1 S_{t-1} + (1-\rho_1) g_t, S_t &= \frac{\hat{S}_t}{1-\rho_1^t} \\ R_t = \rho_2 R_{t-1} + (1-\rho_2) g_t, R_t &= \frac{\hat{R}_t}{1-\rho_2^t} \\ w_{t+1} &= w_t - \frac{\eta}{\sqrt{\hat{R}_t}} \hat{S}_t \end{aligned} \tag{2.8}$$

where ρ_1 and ρ_2 denotes the weight given to the current iteration's gradient in relation to the past history and ρ_1^t, ρ_2^t are used to correct the initializations of R_t and S_t, respectively. Furthermore, η is the global learning rate.

Despite the popularity of adaptive optimization algorithms like Adam, the choice of the optimization algorithm to be used to train deep neural networks is typically based on the familiarity of the user with it, rather than some analytical judgement [16].

2.1.3 Initialization Strategies

The simplest approach to initialize the weights of the network is by setting them to zero. However, by initializing every weight to zero, every neuron will have the same activations, all the calculated gradients will be the same, and consequently, each parameter will suffer the same update. Therefore, it is crucial that the initialization of the weights breaks the symmetry between different units. Drawing from a random Gaussian distribution with mean 0 and deviation 1 would break such symmetry but would also mean that some parameters would have much higher values than others, which would eventually lead to problems like the vanishing or exploding gradients problem. Therefore, different strategies have been proposed to define the distribution from which the initial weights are drawn.

The default initialization strategy for the weights of networks for both the Keras[1] and Tensorflow[2] frameworks is the Glorot's initialization [34] (also called Xavier uniform initialization, because it is the author's first name). In this initialization, the values of the initial weights w of a layer with m inputs and n outputs are to be drawn from the uniform distribution presented at expression 2.9.

$$w \sim U(-\frac{\sqrt{6}}{\sqrt{m+n}}, \frac{\sqrt{6}}{\sqrt{m+n}}) \tag{2.9}$$

For networks using the ReLU activation function, He *et al.* [35] argues that networks should not start in a linear regime, and proposes an initialization method similar to Glorot's initialization. This methodology states that the initial weights w of a layer with m inputs are to be drawn from the uniform distribution presented at expression 2.10.

$$w \sim U(-\frac{\sqrt{6}}{\sqrt{m}}, \frac{\sqrt{6}}{\sqrt{m}}) \tag{2.10}$$

Another type of initialization is the LeCun's initialization [36], which is the default initialization method for the pyTorch[3] framework. It is designed for neural networks with sigmoid activation functions so that the neurons are activated near zero. Otherwise, derivatives may become infinitesimally small, which prevents further training. The values of the initial weights w of a layer with m inputs are to be drawn from the uniform distribution presented at expression 2.11.

$$w \sim U(-\frac{\sqrt{3}}{\sqrt{m}}, \frac{\sqrt{3}}{\sqrt{m}}) \tag{2.11}$$

[1] `https://www.tensorflow.org/guide/keras/sequential_model`
[2] `https://www.tensorflow.org/`
[3] `https://pytorch.org/`

2.2.1 CNN Fundamentals

For image recognition, fully connected feedforward neural networks are not capable of taking advantage of the spatial structure of images. For example, feedforward neural network treat input pixels that are far apart and close together on exactly the same footing. Instead, CNNs or some close variant are used in most neural networks for image recognition problems [28]. They still retain the core concepts of ANNs, but add three different concepts which distinguish them from conventional ANNs (as described by Yann Le Cun [37]):

- 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, image inputs are pixel intensities in a 2D space. Convolutional layers exploit this structure by only connecting neurons to a particular region of the input volume which ensures that the learned filters activate strongly only in the presence of a spatially local input pattern. That region of the input is called the local receptive field and is usually characterized by its square size (*e.g.*, 5×5), and its stride length which can be 1 or more. Note that if a CNN has a 5×5 local receptive field with stride 1 and a 28x28 input image then there will be 24x24 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, and the map from the input layer to the hidden layer is called a feature map. Moreover, a feature detected by a hidden neuron is some kind of input pattern that will cause the neuron to activate. To do image recognition with a CNN, one must use multiple feature maps to recognize different 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 an image. For each feature map, it works by sliding a filter of size $l \times l$ with a stride S and computing the maximum value for the selected part of the input (see example with 2×2 filter and $S = 2$ stride in Figure 2.2). Overall, this concept reduces the number of free parameters, while also not introducing new parameters since max pooling is a fixed function of the input. Consequently, the memory footprint and computation in the network are also reduced [28].

As a result of these three concepts, the overall architecture of CNNs becomes quite different from fully connected neural networks but the same basic concepts still apply. More specifically, the main objective is still to train the weights and biases of the network such that it has good performance classifying the inputs. Additionally, previous activation functions, optimization methods, and initialization strategies can also be applied just like a normal ANN.

Figure 2.3 illustrates a simple CNN architecture proposed by Michael Nielsen to classify numbers from 28x28 pixel images. These images belong to the MNIST which was first

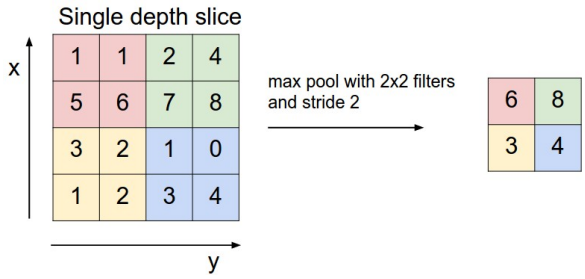

Figure 2.2: Example of max pooling operation with a 2×2 filter and a stride of 2 [38].

presented by LeCun *et al.* [36] and is a large database of handwritten digits. This architecture has 28x28 input neurons used to encode the pixel intensities, then is followed by a convolutional layer using a 5×5 local receptive field and 3 feature maps which results in 3x24x24 feature neurons. Next, a max-pooling layer is applied with 2×2 regions of stride 2 which results in 3x12x12 hidden feature neurons. Finally, the last layer is a softmax layer and 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.

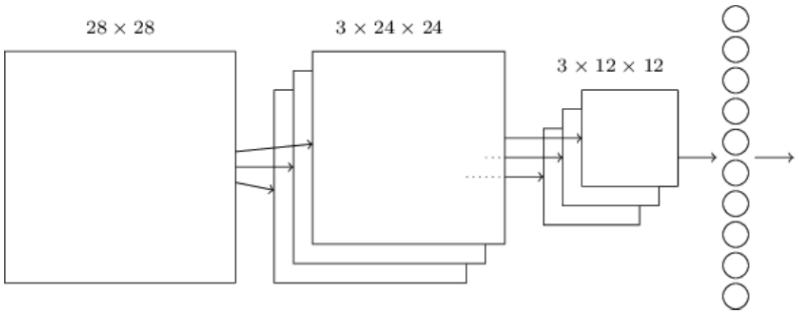

Figure 2.3: Custom CNN architecture used to classify digits from the MNIST dataset taken from Michael Nielsen [28].

This represents a simple CNN architecture for a relatively simple problem. However, more recent CNN architectures used in large-scale data analysis problems (*e.g.*, ResNet [39]), are often more popular because of their state-of-the-art performance on different benchmarks for object recognition tasks, such as ImageNet Large Scale Visual Recognition Competition (ILSVRC). However, they are still derived from the same concepts of this simple architecture, namely, using convolution layers for feature detection, pooling layers for knowledge aggregation, and a classifier containing fully connected layers and softmax layers.

2.2.2 State-of-the-art CNN Architectures

Over the years several CNN architectures have been developed and tested against benchmark challenges such as the ILSVRC [40]. In 2012, Krizhevsky *et al.* [30] submitted for the first time a CNN architecture called AlexNet, which outperformed hand-crafted feature learning methods on the ImageNet dataset. It represented a significant breakthrough that leads to an increasing interest in deep learning methods for visual recognition and classification [41]. It

contained 8 layers, more specifically, 5 convolutional layers (some of them followed by max-pooling layers), and 3 fully-connected. This architecture also used dropout as a regularization method and the ReLU activation function, which showed improved training performance over tanh and sigmoid [30]. This laid the foundation for traditional CNN architectures.

Following AlexNet's main ideas, the VGGNet [42] was created and became quite popular by winning the 2014's ILSVRC. This architecture proved that representation depth is beneficial for the classification accuracy, as it used a traditional CNN architecture but with increased depth along with smaller receptive fields. There are some public variations of this network, one of which has 16 weight layers (VGG16) displayed in Figure 2.4. This architecture is composed of multiple blocks composed of convolutional layers followed by pooling layers that progressively build more abstract features. Furthermore, this architecture employed 3×3 convolutions, 2×2 max-pooling with stride $S = 2$ which meant that resolution is halved after each of these blocks. Finally, the last layer of this architecture has a fully connected structure to convert the results of the convolution into a label.

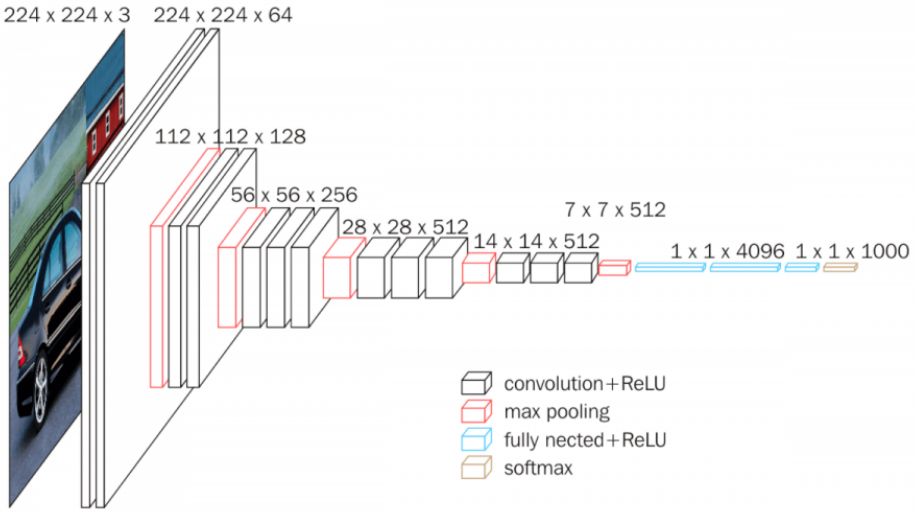

Figure 2.4: Architecture of the VGG16 CNN [42].

Until this point, both AlexNet and VGGNet use plain networks (*i.e.*, networks that stack convolutional layers followed by fully connected layers) and both approaches explore the idea of creating deeper networks in order to obtain better performance. However, as one increases a network's depth, the problem of vanishing gradients starts to be prominent. 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 ILSVRC, and later submitted a paper to the International Conference on Computer Vision and Pattern Recognition [39]. 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 connection (see Figure 2.5).

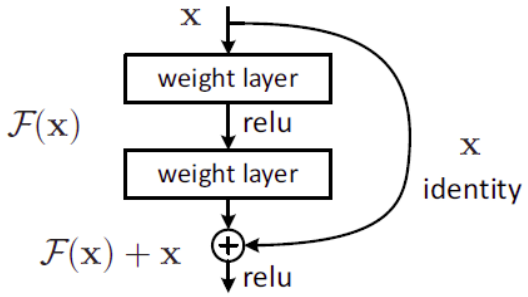

Figure 2.5: Skip connection, the building block of residual neural networks. Taken from He *et al.* [39].

Consider a network with L layers, where each layer of the network implements a non-linear transformation $H_l(\cdot)$, which can be seen as a composite function of operations such as convolutions, batch normalization, activation functions, or even a pooling functions. ResNets have a skip-connection that bypasses the non-linear transformations with the identity function, like the following:

$$x_l = H_l(x_{l-1}) - x_{l-1} \tag{2.12}$$

where x_0 is the input image passed through the network and x_l is the output of the lth layer.

The advantage of this mapping is that the gradient can flow directly through the identity function from later layers to earlier layers, which ultimately helps with the vanishing gradient problem. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlying mapping.

More recently, Huang *et al.* [43] pointed out that whenever 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 network architecture 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 improve information flow efficiency and increases variation of subsequent layers. As such, in DenseNet the lth layer receives the feature-maps of all preceding layers. Equation 2.13 mathematically shows this concept.

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{2.13}$$

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

Moreover, if each function H_l produces k feature maps, then the layer l has $k * (l-1)$ input feature-maps, where k_0 is the number of channels in the inputs layer (also called growth rate). Figure 2.6 visually illustrates this concept, with a 5 layer dense block and a growth rate k of 4. As seen, each layer takes all preceding feature-maps as inputs.

The objective of the growth rate is to regulate how much new information each layer should contribute to the global state of the network. Moreover, once the global state is written

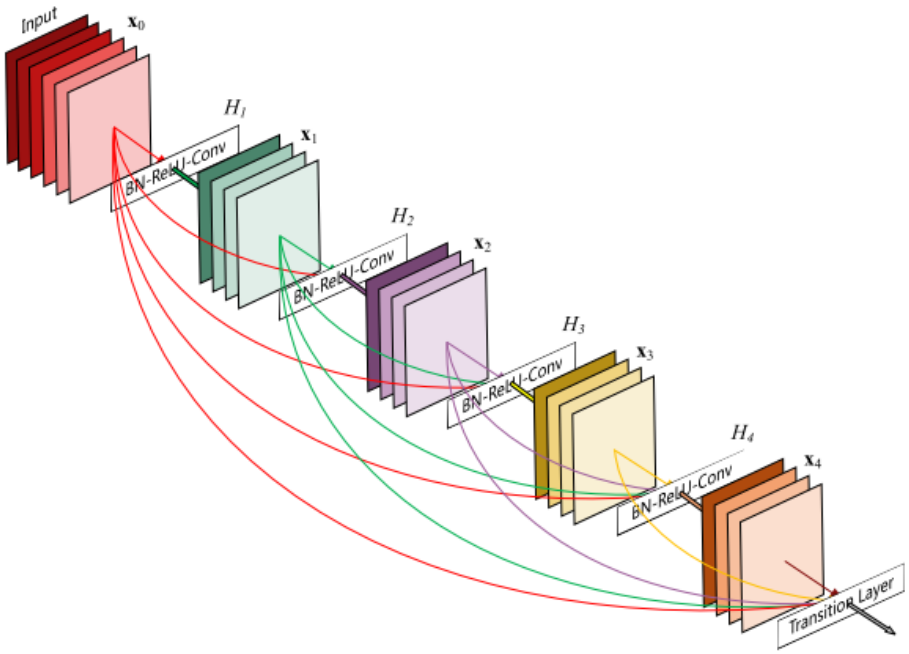

Figure 2.6: A block from DenseNet with five layers each with an expansion of 4. Each layer combines all preceding feature maps through concatenation [43].

by a layer, it can be easily accessed by all the subsequent layers of the network. Therefore, contrarily to traditional neural network architectures, where there is a need to replicate the already acquired knowledge from previous layers, DenseNet is able to re-use knowledge obtained from earlier layers.

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 [44];
- Depth scaling controls how many layers the model has (Figure 2.7 c). All the former presented models explore the idea of scaling up the network in order to capture richer and more abstract features. However, issues such as the vanishing gradients problem start to be more prevalent in deeper networks. Architectures like ResNet [39] and DenseNet [43] attempt to solve this problem, but performance gains diminishes as models become deeper;
- Resolution scaling corresponds to the increase of resolution of input images (Figure 2.7 d). The intuition behind this type of scaling is that the network can see more detail and therefore capture more fine-grained patterns.

Tan *et al.* noted these three scaling dimensions 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 larger images. The authors also argue that one should increase the network width in order to capture more fine-grained patterns, because the images have more information as an input (more pixels). Following this ideas, they proposed a new family of models called EfficientNets [44] which coordinates and balances different scaling dimensions together (see Figure 2.7 e) through the compound scaling method presented by the following expressions:

$$\begin{aligned} \text{depth: } & d = \alpha^{\theta} \\ \text{width: } & w = \beta^{\theta} \\ \text{resolution: } & r = \gamma^{\theta} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ \text{s.t. } & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{2.14}$$

where θ is a coefficient that controls how many resources are available for model scaling and α, β, γ specify how to assign these extra resources to network width, depth, and resolution respectively. All of these parameters can be optimized using grid search.

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

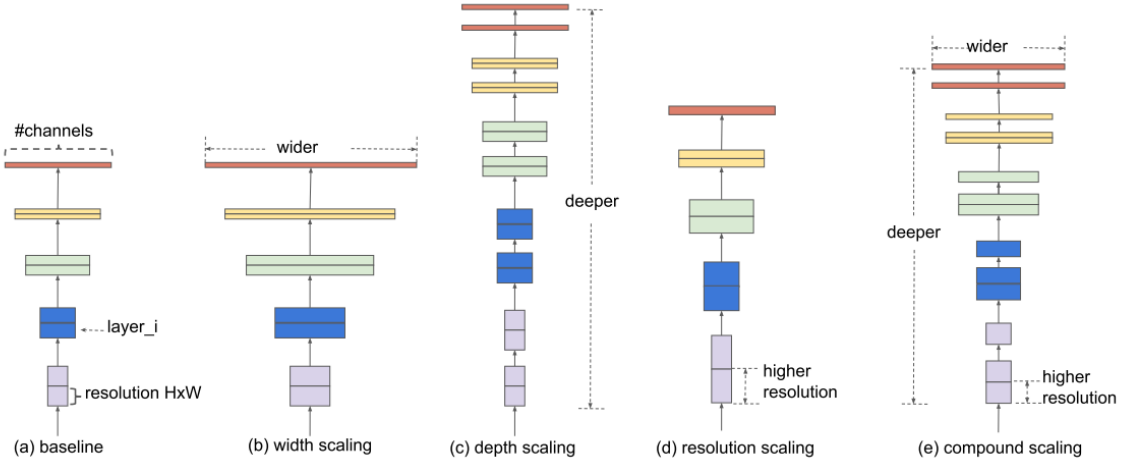

Figure 2.7: Model scaling methods as presented by Tan *et al.* [44].

Other CNN architectures also apply similar concepts for maximizing performance, such is the case of the Inception family of CNN architectures. The InceptionV1 [45] (also called GoogleLeNet) was the first version of this family architecture, and explored the idea of having multiple convolutional operations but with different kernel sizes in the same layer such that different types of information are extracted in a specific layer (*i.e.* information related to

smaller and larger portions of the image). Therefore, the authors designed an inception module with one big filter of kernel size 5×5 for globally distributed information in the image, a medium filter of size 3×3 for slightly smaller information, a 1x1 filter for even smaller information and a 3×3 pooling layer (see Figure 2.8). Moreover, to make the network less computational intensive, the authors limit the number of input channels by adding an extra 1x1 convolution with a max-pooling layer before the 3×3 and 5×5 convolutions. Finally, the InceptionV1 architecture was built with 9 of such inception modules stacked on top of each other (22 layers).

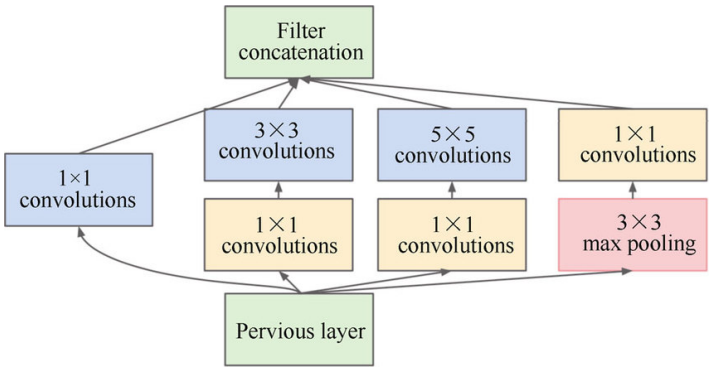

Figure 2.8: Inception module from the InceptionV1 architecture [45].

However, like He *et al.* [39], the authors argued that this network structure is victim of the vanishing gradients problem because it is too deep. Therefore, the authors introduced two auxiliary classifiers with softmaxes that calculated an auxiliary loss over the same labels. Finally, the total loss function is the weighted sum of the auxiliary classifiers (0.3 in the original paper) and the normal classifier. This gave rise to a far more complex architecture than ResNets or VGGNets.

The InceptionV1 architecture was later revised by Szegedy *et al.* [46], which introduced the factorization concept in an attempt to decrease computational complexity. Factorization is a concept in which convolutions with big kernel sizes like 5×5 that are 2.78 times more expensive than 3×3 convolutions, can be reduced into the latter by having two 3×3 stacked convolutions instead. Another option is to factorize convolutions of filter size $n \times n$ to a combination of $1 \times n$ and $n \times 1$ convolutions. For example, a kernel size of 3×3 is the same as first performing a convolution of 1×3 and then perform a 3×1 convolution on the output. Taking this into account, the inception module was redone to accommodate the complexity reduction using this factorization method, which gave rise to 3 different types of inception modules called A, B and C (see Figure 2.9). Furthermore, InceptionV2 overall improved the performance when compared with InceptionV1 (see Table 2.1).

In the same paper as the InceptionV2 approach [46], Szegedy *et al.* also proposed improvements towards the InceptionV2 architecture, which they called the InceptionV3 architecture [46]. These improvements include using the RMSProp Optimizer, 7×7 factorized convolutions, batch normalization [47], and label smoothing (*i.e.*, a regularization technique added to the loss formula).

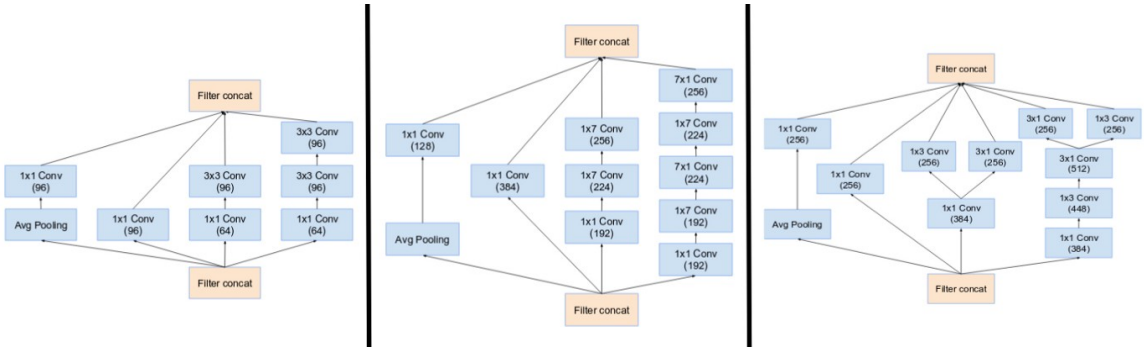

Figure 2.9: Inception module A (left), B (middle) and C (right) of the InceptionV2 architecture [46].

Moreover, Szegedy *et al.* revised the InceptionV3 architecture again with the premise of simplifying the modules of this architecture. This architecture is called InceptionV4 [48], which modified the initial set of operations performed before introducing the Inception blocks. Furthermore, the authors introduced the idea of "Reduction Blocks", which are used to change the width and height of the grid.

In the same paper, inspired by the concept of skip connection of ResNets [39], Szegedy *et al.* proposed a hybrid inception module which integrated the skip connection concept into Inception-ResNetV1 and Inception-ResNetV2 into the inception modules A, B and C (see Figure 2.10). Inception-ResNet-V1 is similar to InceptionV3 and Inception-ResNet-V2 is similar to InceptionV4, in terms of computational complexity, as they use different hyperparameters to adjust that complexity.

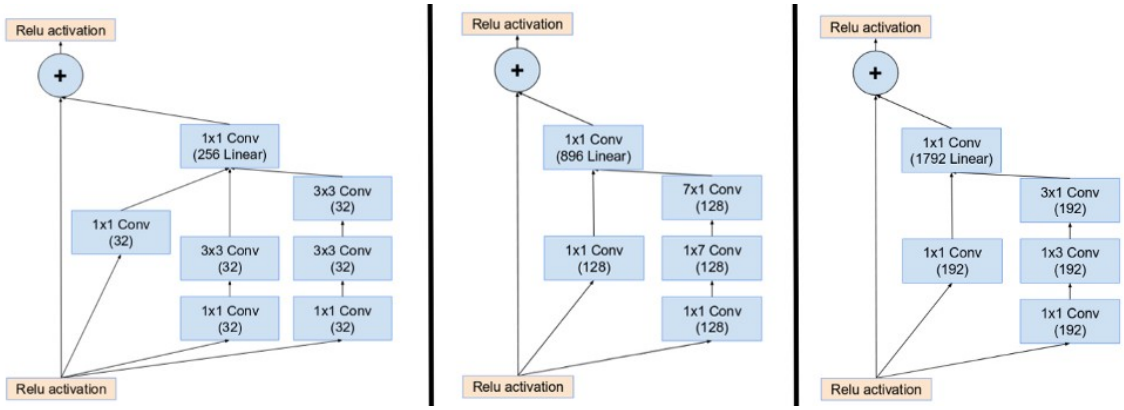

Figure 2.10: Inception module A (left), B (middle) and C (right) of the InceptionResNet (V1 and V2) architecture [48].

Each of these architectures has been tested against benchmark datasets like ImageNet and their performance can be compared in Table 2.1. One can observe that more recent architectures (*e.g.*, EfficientNets) have significantly better accuracy scores in comparison to older ones. Furthermore, over the years, model depth and input size progressively increased, while keeping a relatively low number of trainable parameters and model size.

Model	Year	Size	Top-1 Accuracy	Top-5 Accuracy	Params (Millions)	Depth	Input Size
AlexNet [30]	2012	238 MB	0.570	0.803	≈ 60	8	256x256
VGG16 [42]	2014	528 MB	0.713	0.901	≈ 138	16	224x224
VGG19 [42]	2014	549 MB	0.713	0.900	≈ 143	19	224x224
ResNet50 [39]	2015	98 MB	0.749	0.921	≈ 26	50	224x224
ResNet101 [39]	2015	171 MB	0.764	0.928	≈ 45	101	224x224
ResNet152 [39]	2015	232 MB	0.766	0.931	≈ 60	152	224x224
ResNet50V2 [49]	2016	98 MB	0.760	0.930	≈ 26	50	224x224
ResNet101V2 [49]	2016	171 MB	0.772	0.938	≈ 45	101	224x224
ResNet152V2 [49]	2016	232 MB	0.780	0.942	≈ 60	152	224x224
DenseNet121 [43]	2016	33 MB	0.750	0.923	≈ 8	121	224x224
DenseNet169 [43]	2016	57 MB	0.762	0.932	≈ 14	169	224x224
DenseNet201 [43]	2016	80 MB	0.773	0.936	≈ 20	201	224x224
InceptionV3 [46]	2015	92 MB	0.779	0.937	≈ 24	159	299x299
InceptionResNetV2 [48]	2016	215 MB	0.803	0.953	≈ 56	572	299x299
EfficientNetB0 [44]	2019	5.3 MB	0.773	0.935	≈ 5	NA	224x224
EfficientNetB1 [44]	2019	7.9 MB	0.792	0.945	≈ 8	NA	240x240
EfficientNetB2 [44]	2019	9.2 MB	0.803	0.950	≈ 9	NA	260x260
EfficientNetB3 [44]	2019	12.3 MB	0.817	0.956	≈ 12	NA	300x300
EfficientNetB4 [44]	2019	19.5 MB	0.830	0.963	≈ 19	NA	380x380
EfficientNetB5 [44]	2019	30.6 MB	0.837	0.967	≈ 30	NA	456x456
EfficientNetB6 [44]	2019	43.3 MB	0.842	0.968	≈ 43	NA	456x456
EfficientNetB7 [44]	2019	66.7 MB	0.844	0.971	≈ 66	NA	600x600

Table 2.1: Models comparison on the ImageNet dataset. This comparison is made based on top-1 and top-5 accuracy scores towards the ImageNet validation dataset and entries are chronologically ordered.

2.3 Transfer Learning

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

- Large amounts of labeled 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 hyperparameters and follow heuristics to achieve a good model on the cross-validation process.

However, these requirements can be quite difficult to attain especially for small teams with limited monetary and human resources. Particularly, in the medical imaging context, public datasets to train deep networks are rather scarce [50]. 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.

Transfer learning emerged as a way of relaxing the need for the aforementioned requirements. Transfer learning is a method of reusing a pre-trained model's knowledge for another related task [51]. Using transfer learning means to carry the parameters from a model trained on a generic dataset (*e.g.*, ImageNet), and leveraging it to re-train the model for a different purpose. Moreover, a pre-trained model is a model that was already trained in some domain and can be somehow adapted to a similar domain. For example, if one needs to recognize cats or dogs in an image, instead of training the VGG16 architecture from scratch, one can use the VGG16 pre-trained model (on ImageNet) and leverage the previous weights for the new recognition task.

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 (*i.e.*, to not allow parameter updates while training) some or no portion of those layers.

In order to repurpose the knowledge from a pre-trained model, the classifier must be replaced by a classifier that fits one's needs, while keeping the rest of the architecture (called the convolutional base) intact. Figure 2.11 illustrates three different strategies to repurpose a pre-trained model:

- Strategy 1: Fine-tune the whole model. This means to extract all weights from the convolutional base and fine-tune them along with the classifier's weights. It often requires both more computational power, because it has more parameters to train, and more data to adapt the model to its 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 choose up to which layer should the model be frozen, depending on how much different the problem is compared to the original pre-trained model's problem. Some general advice can be given about this type of strategy. If the dataset to train the pre-trained model is big and similar to the original dataset then fine-tuning fewer layers (only the last layers) will likely yield better

performance. However, if one has a small target dataset which is fairly different from the original dataset in which the pre-trained model was trained on, then fine-tuning most of the layers (except the first ones) might grant better results;

- Strategy 3: Keep the same 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 one is trying to solve, meaning that the two datasets are similar. This can be a good strategy if the available computational power is low because the only parameters required to train are the ones from the classifier layers.

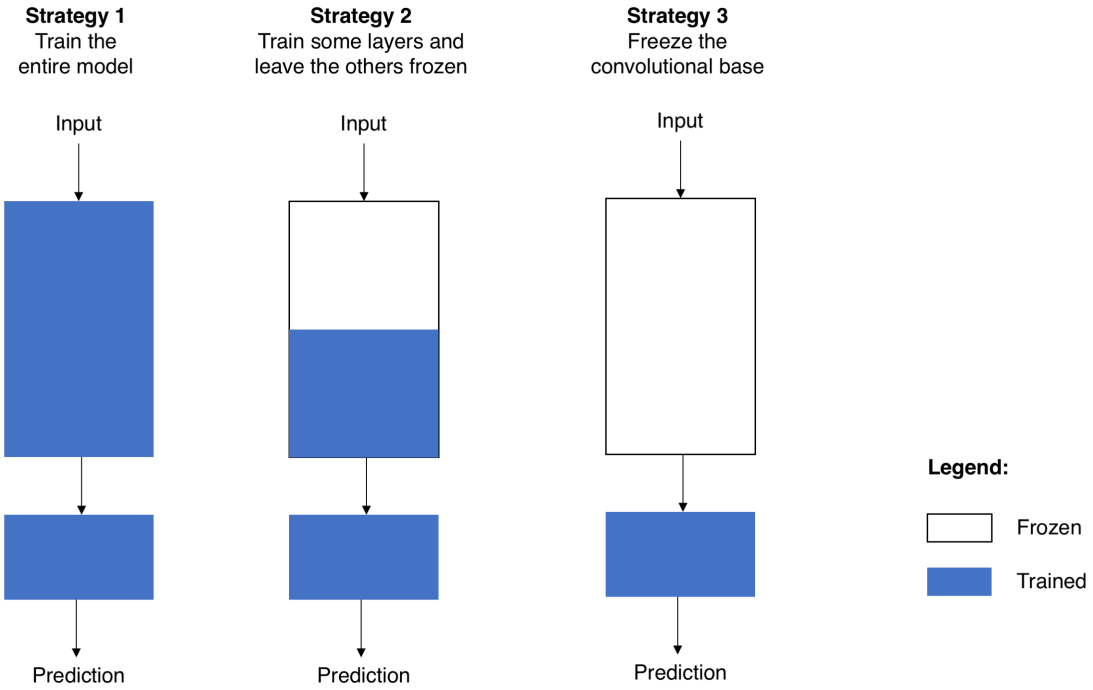

Figure 2.11: Transfer learning strategies as described by Pedro Marcelino [52].

In CNNs, while the convolutional base works as a feature extractor, the classifier uses the features extracted by the convolutional base in order to classify the input. Different options can be used to replace the original classifiers used on the pre-trained models, namely:

- Fully-connected layers followed by a softmax layer, which is the standard approach [30]. The softmax layer outputs the probability distribution over each class label and the classification is equal to the most probable class;
- First proposed by Lin *et al.* [53], one can add a global average pooling layer at the end of the convolutional base, followed by a softmax layer;
- Finally, according to Tang [54], one can improve classification accuracy by training a linear Support Vector Machine (SVM) classifier on the features extracted by the convolutional base.

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 tradeoff is a well-known problem in deep learning that represents a tradeoff 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. One must find a good tradeoff between bias and variance so that the model does not underfit or overfit (see Figure 2.12).

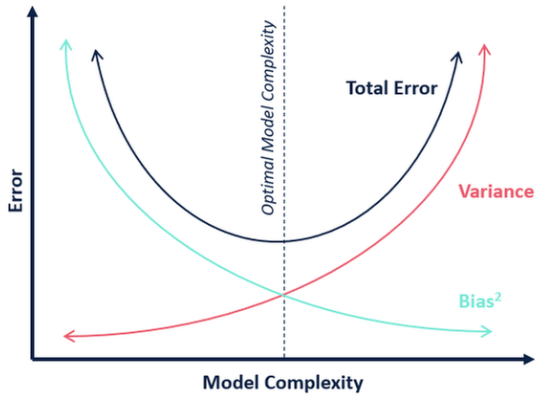

Figure 2.12: Influence of model complexity on the bias and variance. As the complexity of the model rises, the bias of the model decreases but the variance increases. One should find the optimal point of model complexity for a good bias and variance tradeoff, in order to minimize the model's total error.

If the model underfits, then it does not perform well even on the training data, and therefore has high bias and low variance. However, the opposite can also happen, more specifically, producing a model that performs well on the training data but that generalizes poorly to new data [55]. In this case, the model overfits and therefore has a low bias, but very high variance. In order to evaluate whether a model is underfitting or overfitting, one should use metrics that help describe what is happening while training (see Section 2.6).

2.4.1 Underfitting Solutions

A larger neural network can, generally, reduce underfitting because it has more parameters that can be adapted to a specific function and, therefore, minimize the bias of the model (see Figure 2.12). In order to add new parameters to the model, one can increase the depth of the network (more layers) or increase the neurons per layer (wider layers).

An alternative option to deal with underfitting is to increase the number of epochs such that the model eventually finds the optimal trainable parameters to minimize the loss (*e.g.*, increase the early stopping requirements). Finally, one can also reduce the impact of the regularization being applied which would ultimately lead to a better adaptation of the model to the training dataset.

2.4.2 Overfitting Solutions

Multiple solutions to the overfitting problem have been proposed and tested over the years. Presumably the most intuitive is to reduce the network's capacity by removing layers or reducing the number of elements in the hidden layers. This will help because the network will have fewer parameters to learn and therefore minimize the chance of the network memorizing inputs, which leads to a high variance scenario (see Figure 2.12).

Other ways of dealing with this problem are the regularization techniques, which are broadly described by some authors as any method that allows the model to further improve generalization performance [28]. For example, while L1 and L2 regularization attempt to create less complex models [56], techniques such as dropout "reduce complex co-adaptations between neurons" [57], by randomly removing nodes from a specific layer of the network. Alternatively, techniques such as batch normalization can be used to normalize the set of activations in a layer. It works by subtracting the mean from each batch to each activation and then dividing it by the standard deviation from that batch [47].

When a model keeps improving training performance but does not improve validation performance at the end of the training process, it generally means that the model is memorizing the training set rather than learning generalizable features. In this context, early stopping can help reduce overfitting as it stops the model from training further whenever validation performance stagnates for several epochs.

In deep learning, a model is highly dependent on its training dataset in order to achieve good generalization performance [58]. For example, if a dataset does not provide enough proper real-world variation or size or both, then it can easily cause the network to memorize inputs (overfitting). In contrast, a relatively big dataset, representative of the real-world samples, and with enough diversification will force the model to optimize weights towards generalized knowledge rather than memorize inputs.

However, when one does not have a proper dataset available, a concept called data augmentation can be used to alleviate this requirement. The main idea behind this concept is to expand the training data by applying operations that reflect real-world variation to original samples [28]. Shorten *et al.* [58] provides a comprehensive survey about different image processing techniques that can be used for data augmentation, from simple transformations such as translations, rotations, or flips to more complex transformations like distortions or region erasing. Another alternative is to synthetically create new samples through methods like Generative Adversarial Networks (GAN)s, which is a type of neural network.

Moreover, there are two main ways of applying data augmentation for training deep neural networks:

- Offline data augmentation, in which images are generated before the training process and stored in disk for the training of the neural network. This approach can be useful if one wants to employ some kind of schema to apply data augmentation to a particular set of images (*e.g.*, class balance samples). However, it can become troublesome if one has limited disk space, as it increases the storage requirements;
- Online data augmentation, in which images are generated per iteration during training,

and differ in each epoch. In other words, in each epoch, the network is going to be fed a different variation of the original image, rather than the same image. This means that images are created in each epoch iteration and stored in memory for the duration of the iteration and deleted at the end. The big advantage of this approach is related to the lack of requirements for additional disk space to store additional images. However, if the augmentation techniques are rather slow, the training process can take longer. Additionally, it is rather difficult to filter which samples should be augmented according to a schema in this approach.

2.5 Ensemble Learning

Neural networks are non-linear methods, which means that they can learn complex non-linear relationships 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 [59]. As such, depending on the initial conditions, neural networks may learn a very different version of the mapping function from inputs to outputs, which will directly impact the model's performance.

One option to deal with this issue is by using 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 that provides several methods to combine different models, and this work will present a few common ones.

2.5.1 Combine Models Trained on Different Data

If the data used to train each member of the ensemble is different, then different models will be produced and the ensemble will presumably yield better performance. This work highlights two options to combine models with different data:

- Ensemble the models created with k-fold cross-validation: In this procedure, given a dataset with n samples, k different models are trained on k different subsets of the data, with each set of training data having $n - \frac{n}{k}$ samples where the kth set of data is the test set with $\frac{n}{k}$ samples. These models can then be saved and used as members of an ensemble;
- 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.

2.5.2 Combine Models Trained on Different Conditions

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

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 are going to learn substantially different mapping functions between inputs and outputs, which will lower the correlation between predictions.

An alternative to this method is to periodically save the best model during training and then ensemble those models. This method is called snapshot ensembling, described by Huang *et al.* as an optimization process that visits several local minimums before converging to a final solution [60]. A variation for this method is to save models from a range of epochs, perhaps identified by reviewing learning curves [61].

Stochastic gradient descent with warm restarts [62] is another variation in which the optimization procedure is changed during training (*e.g.*, oscillating the learning rate). Afterwards, the best models are saved on specific checkpoints and their predictions are combined.

2.5.3 Combine Predictions from Different Models

The following are some of the methods to combine different model's predictions:

- Average the results from all models, which in neural networks often means to average the softmax probabilities;
- Create a new model that learns how to combine the predictions of the different ensemble models, which is often called model stacking;
- Boosting, in which ensemble members are added one at a time in order to correct the mistakes of prior models;
- Model weight averaging [63], which attempts to average the weights of neural networks with the same architecture, rather than their predictions in an attempt to obtain a slightly more optimized model.

2.6 Performance Metrics

In order to assess whether or not a deep learning model has good train and generalization performance, one should use metrics to quantify the results. In this context, for binary classification one can use the following basic measures:

- True Positive (TP), cases correctly classified as positive;
- True Negative (TN), cases correctly classified as negative;
- False Positive (FP), cases incorrectly classified as positive;
- False Negative (FN), cases incorrectly classified as negative.

One can derive the following metrics from the presented measures:

- True Positive Rate (TPR) (also known as sensitivity or recall) is the percentage of true positives that are classified as positive:

$$TPR = \frac{TP}{TP + FN} \tag{2.15}$$

- True Negative Rate (TNR) (also known as specificity), is the percentage of true negatives that are classified as negative:

$$TNR = \frac{TN}{FP + TN} \tag{2.16}$$

- Positive Predictive Value (PPV), which is also known as precision:

$$PPV = \frac{TP}{TP + FP} \tag{2.17}$$

- Negative Predictive Value (NPV):

$$NPV = \frac{TN}{TN + FN} \tag{2.18}$$

- False Positive Rate (FPR), which is the percentage of true positives that are classified as negative:

$$FPR = 1 - TNR \tag{2.19}$$

For a multi-class classification problem, the above notions can be applied to each label independently to get a per-class metric or be averaged out across all classes, such that a single metric becomes representative of the per-class metrics.

A commonly used metric in the context of deep learning is the accuracy A, also known as the fraction of correct predictions. Consider that there are n samples in a dataset, where $\hat{y}_i$ is the predicted value and y_i is the true value of the ith sample, then the accuracy can be described by:

$$A = \frac{1}{n} * \sum_{i=0}^{n} 1(\hat{y}_i = y_i) \tag{2.20}$$

where $1(x)$ is the indicator function (*i.e.*, a random variable that takes value 1 when the event happens and value 0 when the event does not happen).

However, one should not use the accuracy as the main metric when there is a high class imbalance, because the model can make a lot of correct predictions towards classes with the majority of the samples, classify incorrectly samples from minority classes and still have considerably high accuracy. As such, other metrics that take into account each class's performance as equal such as the Balanced Multi-class accuracy (BMA), should be used:

$$BMA = \frac{1}{C} * \sum_{i=0}^{C} TPR_i \tag{2.21}$$

where C is the number of classes. This metric is the macro-average of the per-class recall, but can also be seen as the accuracy where each sample is weighted according to the inverse pervasiveness of its true class. The BMA score is the same as the accuracy for balanced datasets or when the classifier performs equally well across all classes.

Another frequently used metric is the Area Under Curve (AUC) or the area under the Receiver Operating Characteristic (ROC) curve. A ROC curve is a plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created

by plotting the TPR vs. FPR, at various threshold settings (see example in Figure 2.13). The AUC (also known as AUROC) function summarizes the curve information in one number by computing the area under the receiver operating characteristic (ROC) curve. For multi-class classification problems, the ROC curve can be micro or macro averaged (see example in Figure 2.13) to compute a single AUC score representative of the classifier's performance across all classes.

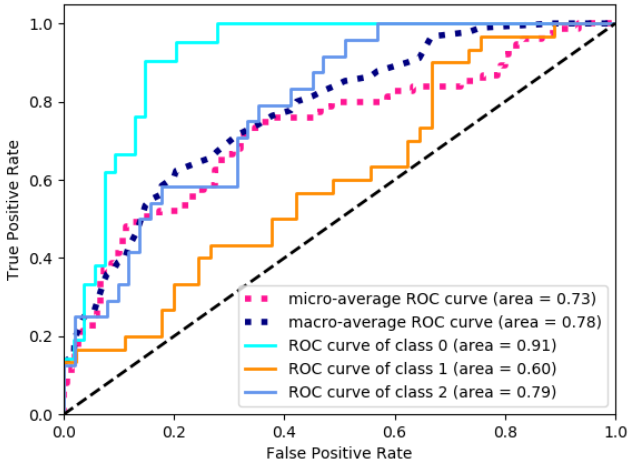

Figure 2.13: ROC (Receiver Operating Characteristic) curve for a multi-class classification problem with micro and macro averages to summarize all classes. Taken from scikit-learn[4].

For all of the above metrics or even more complex ones, the values can be derived from the confusion matrix. A confusion matrix is a tabular way of visualizing the performance, where each entry i, j is the number of observations from group i (true label), but are predicted to be in group j (*e.g.*, left side of Figure 2.14). A confusion matrix can also be normalized, which allows to report ratios instead of counts. To normalize a confusion matrix, one must divide each entry by the sum of the counts per row (*e.g.*, right side of Figure 2.14).

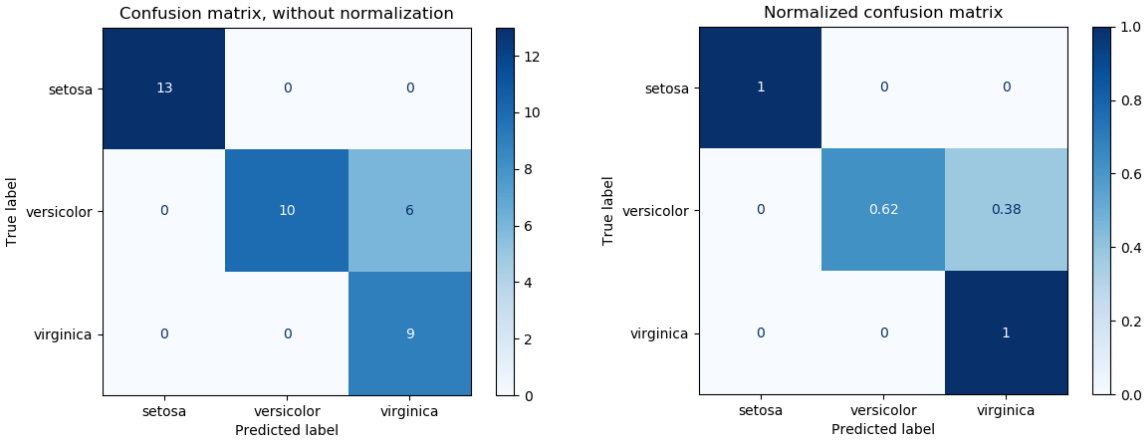

Figure 2.14: Non-normalized vs. normalized confusion matrix for a 3 class classification problem. Taken from scikit-learn.

[4] `https://scikit-learn.org/stable/modules/model_evaluation.html`

2.7 Out of Training Distribution Detection

When machine learning classifiers are employed in real-world tasks, they will often fail if the training and test distributions differ, and, sometimes they provide high confidence predictions while being severely incorrect [64][65]. This is a known problem of deep neural networks which severely limits their adoption in real-world scenarios [66]. Furthermore, this problem has already appeared in the literature in practical classification tasks. For example, Tschandl *et al.* reports that a deep learning model trained to classify skin lesions significantly drops performance when the test data is from a different dataset from the train data [14].

Mathematically, one can frame this problem as: Let P_X and Q_X denote two substantially different data distributions defined on the image space Y. Furthermore, assume that a neural network f is trained with the dataset drawn from the distribution P_X. In this example, one can call P_X the in-distribution and Q_X the out-distribution. In testing, new images are drawn from a mixture distribution $P_{X\times Z}$ defined on $Y \times \{0, 1\}$, where the conditional probability distributions $P_{X|Z=0} = P_X$ and $P_{X|Z=1} = Q_X$ denote in- and out-distribution respectively. Given an image X drawn from the mixture distribution $P_{X\times Z}$, out of distribution detection methods aim to assess whether the image is from in-distribution P_X or not.

A naive approach for detecting out-of-distribution samples is to train the model with a new class composed of samples belonging to a different distribution. However, the extent of the out-of-distribution examples can be too large and it requires the network to be retrained with such samples. Furthermore, it can be difficult to aggregate a set of samples not present in the training distribution which represent that class in a real-world scenario. Moreover, this approach would require a more complex model (*i.e.*, with more trainable parameters) in order to deal with the set of heterogeneous samples present in the out of distribution class. Otherwise, problems related to underfitting could occur which highlights some of the limitations to the practicality of this approach.

In contrast, Hendrycks and Gimpel [66] noted that pre-trained neural networks can be overconfident to out-of-distribution samples even if those samples are irrelevant or unrecognizable. As such, they proposed a baseline method to detect out-of-distribution examples without further re-training networks. The method is based on an observation that a well-trained neural network tends to assign higher softmax scores to in-distribution examples than out-of-distribution examples. In practice, this means that samples should be classified as out-of-distribution if their highest softmax score is lower than a certain threshold. Furthermore, this approach served as a baseline method for future implementations of out-of-distribution detectors.

More recently, Liang *et al.* (a team of Facebook researchers), tried to improve this baseline method with a new detector which they called ODIN [67]. It is based on two main concepts that further increases the gap between the softmax scores of in and out of distribution samples, namely:

- Consider a neural network f trained to classify N classes, such that $f = (f_1, ..., f_N)$. For each input x, the neural network assigns a label $\hat{y}(x) = argmax_i S_i(x; T)$ by computing

the softmax output for each class. Equation 2.22 shows the adjusted softmax.

$$S_i(x;T) = \frac{exp(f_i(x)/T)}{\sum_j^n exp(f_j(x)/T)} \tag{2.22}$$

where $T \in \mathbb{R}^+$ is the temperature scaling parameter and set to 1 during the training. For a given input x, the softmax score (or the maximum softmax probability) is given by $S_{\hat{y}}(x;T) = max_i S_i(x;T)$.

By tuning T, one can increase the gap between softmax scores of in and out of distribution samples in the test set. Moreover, according to Hinton *et al.* adding temperature scaling to the softmax function will distill the knowledge of the neural network [68] as well as provide a way to calibrate the prediction confidence in classification tasks [69];

- Goodfellow *et al.* noted that small perturbations added to samples before prediction tend to decrease the softmax score for the true label and force the neural network to make a wrong prediction [64]. This also reflects into out of distribution detection, as these perturbations can have a stronger effect on the in-distribution images than that on out-of-distribution images. This concept is leveraged by ODIN as a way to further make in and out of distribution samples more separable through the Equation 2.23.

$$\widetilde{x} = x - \varepsilon sign(-\nabla_x log S_{\hat{y}}(x;T)) \tag{2.23}$$

where the parameter ε can be interpreted as the perturbation magnitude.

The ODIN detector works by obtaining the preprocessed image $\widetilde{x}$ according to the Equation 2.23, then feed the result into the neural network and calculate its softmax score $S(\widetilde{x};T)$. Finally, one should compare the obtained score with a pre-defined threshold δ so that images that are part of the in distribution dataset, have a softmax score above the threshold. Otherwise, they are predicted to be an out of training distribution sample. (see Equation 2.24).

$$g(x;\delta,T,\varepsilon) = \begin{cases} 1, & \text{if } max_i p(\widetilde{x};T) \leq \delta \\ 0, & \text{if } max_i p(\widetilde{x};T) > \delta \end{cases} \tag{2.24}$$

where T, ε and δ are chosen so that the TPR (*i.e.*, the fraction of in-distribution images correctly classified as in-distribution images) under some out-of-distribution image dataset is 95%.

Furthermore, custom metrics are usually employed to compare the results attained by these detectors, namely:

- FPR at 95% TPR can be interpreted as the probability that a negative (out-of-distribution) example is misclassified as positive (in-distribution) when the TPR is as high as 95% (used by ODIN [67]).
- Detection Error, measures the minimum misclassification probability over all possible score thresholds (used by ODIN [67]).
- AUROC which depicts the relationship between TPR and FPR. A perfect detector corresponds to an AUROC score of 100% (used by ODIN [67] and Hendrycks and Gimpel method [66]).

CHAPTER 3

Skin Lesion Diagnosis

This chapter provides an overlook into the state-of-the-art applications and methods used to diagnose skin lesions, including hand-crafted image processing techniques and recent end-to-end deep learning approaches. In order to provide some context, Section 3.1 reviews the literature related to dermatology applications for the self-surveillance of skin lesions. Section 3.2 presents an exploratory view into different handcrafted computer vision methods for skin lesion diagnosis. Section 3.3 focuses on end-to-end approaches for skin lesion classification that use deep learning methods. Finally, Section 3.4 identifies some of the limitations and opportunities of deep learning methods towards skin lesion diagnosis.

3.1 eHealth and mHealth Applications

Online health, eHealth, and mHealth applications represent a rapidly developing field of medicine that has the potential to become a powerful tool in the diagnosis and management of skin diseases [70]. 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.

Currently, several production-ready skin lesion classification systems are available for both skin professionals and patients wishing to self monitor their skin. Kassianos *et al.* [71] reviewed mHealth 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 [12]. 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

an 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.* [72] reviewed the literature on mHealth 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 malign 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 same year, Jaworek-Korjakowska and Kleczek [70] 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). SkinVision in particular classifies lesions as either low, medium or high risk of skin cancer by using a risk assessment algorithm based on gray-scale images of lesions and their associated fractal maps. It achieves an overall sensitivity of 73%, a specificity of 83%, and an accuracy of 81%. The positive and negative predictive values were 49% and 83%, respectively [70].

As an alternative, some web applications help dermatologists at the diagnosis of skin lesions, such as Metaoptima's Dermengine web application [73]. Their Visual Search tool compares user-submitted images with similar images in a database of thousands of pathology-labeled images gathered from other dermatologists, based on visual features such as the color, shape, or patterns of a lesion [73].

Unfortunately, from the presented eHealth applications, only two were certified by authorities, namely, SkinVision which received the European "CE" Marking and DermaCompare that was approved by the U.S. Food and Drug Administration [70]. This highlights larger concerns with these types of applications, namely, the lack of the required performance for medical-grade applications and the false advertising of these systems performance. This is corroborated by reports such as the one done by Maier *et al.* for the Journal of the European Academy of Dermatology and Venerology, which shows that there is a lack of accuracy within mHealth apps such as SkinVision on tasks like melanoma detection [74]. Other incidents also

reflect this concern, such as in 2015 the Federal Trade Commission in the U.S., filing cases against several skin mole evaluation applications for false advertising of their capabilities to detect skin cancers [75].

3.2 Hand-crafted Image Processing Methods

As seen in the previous section, some of the skin lesion classification eHealth/mHealth applications used hand-crafted image processing methods in order to make assessments (*e.g.*, ABCDE method). Typically, to classify a skin lesion from dermoscopic images using handcrafted features, 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 [9]. Different approaches for each of these tasks will be presented.

3.2.1 Lesion Segmentation

Skin lesions can be distinguished 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 so that further processing can be applied (*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 dermoscopic images often come with various artifacts (*e.g.*, hairs, ruler marks, date markers, ink, bubbles), and in many cases, there is a low-contrast between the bounded region and the surrounding area (*e.g.*, lighting conditions, skin tone variations). 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. Histogram thresholding is one simple example used by some authors like Celebi *et al.* [76], which segments image regions based on whether pixels from that region are below a specific pixel intensity or not. Edge detection is another simple technique used [77] in which lesions are segmented based on discontinuities. However more complex approaches like clustering [78], active contours [79], graph theory [80] and probabilistic modeling [81] can also be found in the literature.

3.2.2 Feature Extraction

With the bounded region, one can start feature extraction by identifying points of interest within that region. Such interest points can be attributes such as color, texture, shape, structure, relative size, and location of the lesion [82]. 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 [83][84] 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 [12] evaluates melanoma diagnosis using four criteria which include asymmetry, border irregularity, color, and diameter. Each criterion is multiplied by a weight factor to establish a Total Dermoscopic Score (TDS). A lesion is classified as benign if its TDS score is less than 4.75, if it is within the 4.75 and 5.45 range, it is considered as a possible melanoma, and finally if the TDS is above 5.45 it is likely a melanoma;
- Menzies method [85] 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 are radial streamings, multiple brown dots, blue white veils, pseudopods, peripheral ebony globules, depigmentations, multiple hues, numerous blue or dark spots, and finally expanded networks. For a lesion to be classified as melanoma at least one positive feature must be found;
- 7-point checklist [86][87] 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 [11] is designed as a simplified form of pattern analysis in which lesions are evaluated according to the color, architecture, symmetry, and homogeneity.

A comparison of these algorithms reveals two diverging approaches to simplified melanoma detection. The ABCD rule and CASH mainly quantify the overall organization of a lesion by assessing features such as symmetry, architectural disorder, border sharpness and heterogeneity in colors and structures. The 7-point checklist relies on the identification of atypical appearances of dermoscopic structures (*e.g.*, atypical network) in distinction from otherwise normal counterparts or on identifying unique structures strongly associated with melanoma. The Menzies method includes elements of both approaches. Although each algorithm has unique criteria, there is significant overlap in their concepts.

3.2.3 Disease Classification

Finally, the image features extracted can be fed to a classifier that should be able to distinguish between different types of lesions (*e.g.*, melanomas from benign lesions). Such classifiers can be based on a multitude of algorithms such as artificial neural networks or support vector machines. Figure 3.1 illustrates the distribution of the most used algorithms for classifying features from skin lesions as presented by Prabhu *et al.* [9].

While these types of approaches provide a way of classifying skin lesions, they require domain knowledge, particularly, when performing segmentation and feature extraction. More recently, deep learning techniques have become prominent in fields such as medical imaging, and more specifically in skin lesion diagnosis, as seen by some surveys [88] [89]. In contrast

with hand-crafted methods, deep learning can be used without requiring preliminary steps such as lesion segmentation or feature extraction.

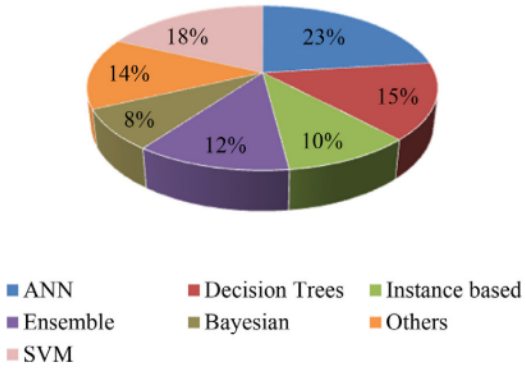

Figure 3.1: Contribution of each classification method to the total number of dermoscopic studies according to the review paper of Prabhu *et al.* [9].

3.3 Deep Learning for End-to-end Classification of Skin Lesions

Deep learning refers to computational models composed of multiple processing layers capable of learning representations of data with multiple levels of abstraction [16]. 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 [90]. 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.

Recently, deep neural networks appear as state-of-the-art solutions for medical imaging problems due to advancements in the field [89]. 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 lastly, the development of high-level modules such as Tensorflow[1] [91] that help train and test neural networks.

One key advantage of deep learning over other machine learning algorithms is that it removes the need for feature engineering, a process that requires knowledge expertise of the problem domain but can be time-consuming and introduce human error if not all the algorithm details are described (*i.e.*, edge cases). Another key advantage of deep neural networks is that these models scale well with the amount of data available, as opposed to other machine learning methods in which performance stagnates past a certain amount of samples [41].

3.3.1 Transfer Learning Approaches

Perhaps one of the most popular approaches for skin cancer classification using deep CNNs, 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,

[1] `https://www.tensorflow.org/`

Edinburgh Dermofit Library, and the Stanford Hospital, totaling an astonishing 129450 samples which remains one of the biggest efforts in data collection in the area. This dataset was also oversampled using data augmentation with flips, rotations, and crops. The authors follow a transfer learning approach by leveraging the weights of the InceptionV3 [46] network pre-trained on ImageNet [18], on top of which they replaced the classifier. Afterward, they measured the network's performance by pitting it against 21 dermatologists on biopsy-proven samples. Finally, the presented results proved that their classifier had comparable performance to that of those board-certified dermatologists with an AUC of 0.94.

Haenssle *et al.* [5] presented a very similar approach to Esteva *et al.* [3]. They also fine-tuned the InceptionV3 pre-trained model [46], but the analysis was limited to samples of melanoma and benign nevi. Similarly to Esteva *et al.*, they compared their approach with 58 dermatologists, which at the time was the largest number of dermatologists involved in a publication about automated skin lesion classification [15]. They achieved AUC of 0.86, which is a considerably lower score than that of Esteva *et al.*.

More recently, Tschandl *et al.* [13] also compared the performance of 95 human raters with different experience in dermoscopy (all medical personnel, from which 62 board-certified dermatologists):

- Their approach employed a combined CNN trained on 13724 samples of lesions (7895 dermoscopic and 5829 close-ups) and tested on 2072 samples. These samples are part of 8 different lesion classes, namely, actinic keratoses and intraepithelial carcinoma (also known as Bowen disease), basal cell carcinoma (all subtypes), benign keratosis-like lesions (including solar lentigo, seborrheic keratosis, and lichen planus–like keratosis), dermatofibroma, melanoma, invasive squamous cell carcinoma and keratoacanthoma, benign sebaceous neoplasms, and benign hair follicle tumors;
- The combined CNN is composed of two distinct pre-trained models, namely the InceptionV3 [46] model and the ResNet50 model [39] pre-trained on ImageNet;
- Human raters were compared to the combined CNN based on the AUC metric calculated from 50 cases drawn randomly from the entire test set. Furthermore, human raters were divided into three groups of experience, namely, beginners, intermediates, and experts.
- Results showed that the combined CNN was able to diagnose lesions as accurately as human experts (0.733 vs 0.741), and outperform both beginner and intermediate experienced raters.
- Although these results are promising, the authors pointed that the evaluation metrics used to measure diagnostic performance might not represent the performance in a real-world scenario, because it does not take into account the potential loss of life-years and apply penalties to misdiagnoses of more aggressive disease.
- Lastly, the authors highlighted the importance of the dataset used to train these models and affirmed that future efforts should be put into making larger amounts of samples of common and rare lesions available to the research community, such that future approaches can increase performance.

For general image recognition problems, datasets such as ImageNet [18] which contains over 14 Million samples with over 20000 classes 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 [15].

However, some attempts have been made to address this problem. For example, the publicly available Human Against Machine with 10000 images (HAM10000) dataset [92] which consists of 10015 dermatoscopic images. It contains classes such as melanoma, melanocytic nevi, actinic keratoses, intraepithelial carcinoma, Bowen's disease, basal cell carcinoma, benign keratosis, dermatofibroma, and finally vascular lesions. Furthermore, organizations such as ISIC provide an open-source public access archive of skin images, called ISIC archive [93]. It can be used for the development or testing of automated skin lesion diagnosis systems [24]. This organization also places a challenge every year which serves as a benchmark for skin lesion classification systems.

International Skin Imaging Collaboration Challenge 2018

Automated skin lesion classification systems should not only provide information and treatment options, but also detect skin cancer cases with reasonable sensitivity and specificity [94]. While the first goal is a multi-class problem (diagnosis out of range of classes), the second is a binary one ("biopsy" or "no biopsy"). In third task of the ISIC 2018 challenge, participants were asked to develop a multi-class classifier to distinguish between 7 different types of skin lesions. The provided dataset include lesion samples of melanocytic nevus, melanomas, basal cell carcinomas, actinic keratosis, benign keratosis, dermatofibromas, and finally vascular lesions. Participants were ranked based on the BMA as it is closer to real evaluation of a dermatologist [94], but other metrics such as accuracy or AUC are computed for scientific completeness (see Section 2.6).

The top 3 submissions had BMAs of 88.5%, 88.2%, 87.1%, respectively, and were all submitted by Aleksey Nozdryn-Plotnicki and Yolland [95], which were part of a team at Metaoptima (the company behind Dermengine [73]). To train those models they used the provided dataset, plus samples from the ISIC archive and samples from a proprietary dataset. All samples were preprocessed by normalizing them with the shades of gray method [96]. Additionally, they augmented the training data by performing random horizontal flips, random rotations, changes in brightness, saturation, and contrast. They employed a transfer learning approach using several models pre-trained on ImageNet (*e.g.*, InceptionV3 or ResNet) and then ensembled the models with the highest accuracy [95].

Furthermore, some submissions presented interesting ways to improve the results without the need for external data. More specifically:

- Gessert et at. [20] performed an extensive hyperparameter tuning with different optimization algorithms, in which they concluded that the Adam optimizer [33] was the best choice. They also have a particularly interesting approach for inference (shown in Figure 3.2), while performing competitively with an BMA of 0.83 (fourth place). They ensembled different pre-trained models, namely, SENet154, ResNeXt101, Densenet201, Densenet161, Densenet169, SE-Resnet101, and PolyNet. 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. For cross-validated models, predictions are combined using a SVM. Finally, both results are combined by averaging the softmax probabilities;

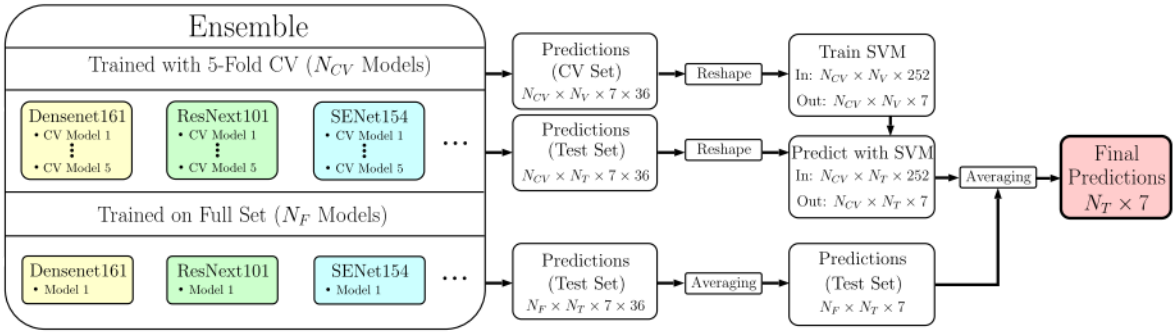

Figure 3.2: The evaluation strategy for the generation of final predictions of Gessert *et al.* [20].

- Ashraful Alam Milton [97] followed an approach based on transfer learning from models of PNASNet-5-Large, InceptionResNetV2, SENet154, InceptionV4 trained on ImageNet. He interestingly noted that in the first few epochs the gradient is very erratic. Thus, he presented an approach in which during the first 2 epochs only the classifier is trained with the convolutional base frozen in order to avoid updating weights towards the wrong direction. Finally, his approach achieved a BMA of 0.76 on the validation set;
- Bissoto *et al.* [98] (who won 3rd place in the 2017 edition) transferred 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.* [14] compared the diagnostic accuracy of 139 machine-learning algorithms from ISIC 2018 with 511 human readers, with the 55.4% being board-certified dermatologists, 23.1% being dermatology residents and 16.2% being general practitioners. Overall, machine learning algorithms outperformed human readers on the large majority of measures. For example, in sets of 30 randomly selected lesions, the best machine-learning algorithms achieved a mean of approximately 8 more correct diagnoses than the average human reader and a mean of approximately 7 more correct diagnoses than expert readers. However, the authors advocated that higher accuracy does not necessarily mean better clinical performance or patient management. For example, these algorithms were trained to optimize 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 training distribution samples). The authors argued that this is a possible limitation of deep learning based methods which needs to be addressed in future research. Other authors such as Han *et al.*, also noted this same limitation in their study [27].

International Skin Imaging Collaboration Challenge 2019

As a response to the concerns presented by Tschandl *et al.* [14] and Han *et al.* [27], the ISIC 2019 challenge asked participants to classify dermoscopic images among nine different diagnostic categories, with 8 known classes, (melanocytic nevus, melanoma, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, and finally squamous cell carcinoma) and one "unknown" class which conceptually means none of the other classes. Therefore, the "unknown" class is the equivalent to the detection of out of training distribution samples. Similarly to the third task of the ISIC 2018 challenge, participants could use private data to improve the network's performance and were ranked based on the BMA score [24].

The best submission towards the ISIC 2019 challenge (images only) was done by Gessert *et al.* scoring 0.636 BMA [21]:

- In addition to the challenge's dataset, 995 dermoscopic images from the 7-point dataset [99] and 1339 images from an in-house dataset were used for training. The in-house dataset also contained additional images that were used to train an outlier class to deal with the out of training distribution samples present in the test set;
- Preprocessing is done by cropping the images, performing image binarization, then applying the shades of gray color constancy method [96], 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 holes [100];
- 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 [33] and a weighted cross-entropy loss function to deal with class unbalance;
- 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;
- Predictions for each model are made based on which input strategy was used. The final prediction is made using an ensemble of the best performing models.

At second place, Zhou *et al.* [25] achieved a BMA of 0.607 with an arguably simpler approach to deal with the out of training distribution samples for this challenge:

- Unlike Gessert *et al.* [21], no additional data was used for training and testing. Images were resized to the target model's input size, but maintained the aspect ratio. Furthermore, they used the shades of gray color constancy method [96] to normalize samples;

- Images were augmented using random cropping, scaling, rotations and color transformations;
- Like most submissions, they used a transfer learning approach by fine-tuning Densenet121, SE-ResNeXt-50, SE-ResNeXt-101, EfficientnetB2, EfficientnetB3, and EfficientnetB4 which were all pre-trained on ImageNet. Models were trained for 90 epochs with the Adam optimizer [33]. Furthermore, they used a learning rate of 5^{-5} which was 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 to optimize the BMA values, but for the final submission EfficienetB3, EfficienetB4 and Seresnext101 were used;
- During inference, for each test image, either 16 or 25 ordered same-sized patches are fed into each network and the final prediction is obtained by averaging the prediction probabilities of these patches;
- Finally, the unknown samples are handled by simply classifying the images whose top-1 probability is less than 0.35 as "unknown".

In third place, Pollastri *et al.* [26] achieved a BMA score of 0.593 and opted for a more complex method of 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 to rescale them to a square size of 512x512 pixels. Pixel intensities are normalized to have the mean equal to 0 and standard deviation equal to 1;
- Training is performed using a cross-entropy loss function which is weighted according to the inverse prior probability of each class;
- For baseline model A, DenseNet201, ResNet152 and SE-ResNeXt-101 were the pre-trained models used, while for baseline model B, SE-ResNeXt-50, SE-ResNeXt-50 and the SeNet-154 were the preferred models;
- Both baseline models use online data augmentation. More specifically, baseline model A uses operations such as rotations, flips, Gaussian blur, adding Poisson noise, gamma contrast changes, cutout holes [100], and 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. Furthermore, test samples go through the same augmentations as the ones performed during training. In contrast, for baseline model B the snapshot ensemble technique [60] was used;
- Their approach to determine the unknown class is to use the Out of Distribution Detector by [101]. Eight CNNs 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. The inference is performed in a voting scheme where each network votes whether a given sample is part of the training set or not.

Hsin-Wei Wang's approach [102] is particularly noteworthy because the materials for replicating his submission are available on Github[2]. His method achieved a BMA of 0.505, a substantially worse performance than the best approaches mainly due to the poor results of the out of distribution detector towards 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 [99] 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;
- Similarly to the best approaches, an ensemble of models was used by averaging softmax probabilities made by three models (pre-trained with the ImageNet dataset): the DenseNet201, Xception, and ResNeXt50. During training, all layers are frozen for weight initialization of the classifier during the first 3 epochs, then all the layers are unfrozen and fine-tuned during 100 epochs with the Adam optimizer [33]. He uses a learning rate of 10^{-5} which is reduced by a factor of 10 if the validation loss has stopped improving for 8 epochs;
- 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 [67], 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).

3.3.2 Learning from Scratch Approaches

Training deep neural networks from scratch for skin lesion diagnosis requires designing the network architecture, tuning the initial weights, and finding the best hyperparameters that best fit this classification problem. As such, this task requires in-depth knowledge and reasoning of deep learning techniques as well as the ability to perform experiments to test different approaches. However, some authors argue that this approach can be beneficial.

For the ISBI 2016 challenge, Yu *et al.* [103] trained a deep residual neural network (with more than 50 layers) from scratch with 900 images from the ISIC Archive to identify melanoma cases. They employed the residual learning technique used by ResNets [39] called skip connections, which as demonstrated in Section 2.2.2, helps with the overfitting problems of deeper neural networks. Additionally, they used batch normalization [47] to deal with the vanishing gradients problem [104]. This approach is composed of two stages: (i) The first

[2] `https://github.com/wanghsinwei/isic-2019`

stage uses a CNN to segment the lesion. (ii) The second stage uses another CNN with a softmax layer at the end and a SVM, which is used to classify the lesion between melanoma and non-melanoma. This approach ranked first in the ISBI 2016 classification challenge and second in the segmentation challenge, which was according to the authors due to the benefits of using a deeper network in comparison with other approaches that used architectures like VGG16 [42] or AlexNet [30].

More recently, in 2019, Ly *et al.* [105], trained multiple models from scratch intending to deploy such models for offline usage in smartphones. They justified the decision of training a model from scratch by arguing that using pre-trained models with large neural network architectures requires a lot more parameters when compared with models trained from scratch. This ultimately would require the smartphones to have much higher computational requirements (*e.g.*, memory), which are not available on such mobile environments. The authors argued that their approach provides competitive results (86% accuracy) in comparison with other transfer learning based approaches, while having much more compact model (approximately 29 Megabytes). 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.

3.4 Challenges and Opportunities of Deep Learning Methods

Despite the promising results obtained by using deep learning techniques towards skin lesion classification, several challenges are facing these approaches that remain unsolved. In turn, these challenges present new exciting opportunities to improve the current state-of-the-art methods.

3.4.1 Interpretability

A common challenge for deep learning is the interpretability of the models and their predictions [17]. To interpret these nonlinear mathematical models is key to risk-averse tasks like skin lesion diagnosis and not doing it would result in distrust in the model's predictions [17]. Additionally, a model that achieves state of the art performance on a particular task could have identified patterns that would be useful for medical personal to fully understand. Moreover, these models can be susceptible to adversarial examples and output high confidence scores for samples that resemble noise [17]. Therefore, it is important to understand a model's output to address these types of situations.

Currently, understanding the way neural networks learn is an active area of research towards medical imaging, particularly, for skin lesion classification. For example:

- Van Mole *et al.* [106] proposed an self-explainable model which allowed the visualization of the filters learned by the CNN. Their results showed that the CNN was able to learn filters that were sensitive to: border, lesion and skin color, hair, and artifacts. However, this approach only allows for human inspection during the inference phase, and although it does improve explainability, it has no impact on the performance of the network.

- More recently, Barata *et al.* [107] presents an approach towards the ISIC 2017 and ISIC 2018 challenges to address the lack of explainability of state of the art approaches by using a multi-task network to perform a hierarchical diagnosis of skin lesions. The obtained results on the ISIC 2017 and ISIC 2018 datasets showed that the model can identify clinically relevant regions in the images, which the authors argued that could be used to provide an explainable diagnosis.

These approaches demonstrate that deep learning for skin lesion classification can provide self-explainable results with interpretability capabilities for human assessment. However, further research should be done to improve these methods.

3.4.2 Data Limitations

As demonstrated in Table 2.1, CNN architectures can have Millions of trainable parameters which need to be properly optimized. The basis to achieve this goal is the availability of a huge amount of data [50]. However, if there is a lack of data available which is often the case for medical imaging problems like skin lesion diagnosis [108], these models will ultimately be more susceptible to problems like overfitting. It is even more relevant when there is a level of uncertainty in the labels of the training examples (*e.g.*, labels attributed based on visual examination rather than biopsy proved lesions), which introduces mistakes for the model to learn. This problem can be accredited to the lack of standards and rules that dictate how samples and labels should be attributed. As such, to not deal with this issue will ultimately undermine the quality and usefulness of deep learning based automated skin lesion diagnosis systems.

Therefore, organizations like ISIC attempt to tackle this concern with a public archive of labeled skin lesions for the teaching and development of automated skin lesion diagnosis systems. They propose standards related to technologies, techniques, and terminologies for skin imaging, to overcome the lack of benchmarks that set the ground truth for comparing results within the research community. This archive keeps growing with more and more datasets being uploaded, which will ultimately ease the requirement for large datasets and help to advance the knowledge of this field.

However, there are other alternatives to solve the data limitations. Several approaches towards skin lesion diagnosis attempt to alleviate the lack of data through synthetically generating samples (*i.e.*, data augmentation). For example, an image can easily be rotated, flipped, or translated and retain its label. This can greatly expand the number of training examples but artificially treats such derived images as independent instances and sacrifices the structure inherent in the data [17]. Several cautions should also be taken into consideration, namely the test and validation sets should be composed of original samples such that there is no bias in the performance results. This can ultimately lead to a model that easily discriminates synthetic examples but does not generalize the attained knowledge to real data [17]. Furthermore, one should carefully consider which augmentation techniques makes sense in the context of skin lesion classification. Overall, data augmentation can be a valuable concept to overcome the lack of samples, but one that requires more rigorous evaluation.

3.4.3 Out of Training Distribution Test Data

When one wishes to evaluate the performance of a deep learning model, the commonly used procedure is to use metrics calculated over a test set in order to attain an expected performance for the model in the real world. However, if the test samples are from a very divergent data distribution from the one met in the real world, then there is going to be substantially lower performance than expected (see related work by Zech *et al.* [109]).

Publicly available datasets such as the HAM10000 [92] often have high-quality samples that were previously filtered by skin professionals. However, in the clinical world, the test data can be quite different from those datasets, which can become a major obstacle when deploying these deep learning systems into a production environment. Therefore, it is crucial to create automated skin lesion diagnosis systems with the capability of flagging samples that are not part of the original training distribution for further diagnosis of a physician.

Out of distribution detection is a term often used in the literature to describe methods that attempt to identify samples not part of the original training distribution (see Section 2.7). In the context of deep learning for skin lesion diagnosis, out-of-distribution detection can be referred as systems that are capable of identifying lesions which either there was insufficient training data to render a reliable decision for such sample, or for which no training data was used to train a classifier with that category of lesions [108].

Efforts have been made to point out of distribution detection as a research focus. More specifically, Tschandl *et al.* proposed that the ISIC challenges should address this problem after reviewing ISIC 2018 submissions [14]. Afterward, the ISIC 2019 challenge includes an "unknown" class, which in practice means that a lesion is "unknown" if it is not part of the original training distribution. As denoted in Section 3.3, several approaches attempted to tackle this issue, but questions remain about their effectiveness in the domain of skin lesion diagnosis. Therefore, comparing these different approaches could be an interesting research focus.

3.4.4 Hardware Limitations

Training deep neural networks is costly requiring both time, energy, and memory. One of the reasons deep learning methods took off as state of the art methods in multiple domains is due to the use of GPUs in the training process. GPUs have a parallel nature capable of accelerating the mini-batch gradient descent and excel at the matrix and vector operations so central to deep learning [17]. However, these processors have bounded memory which can in some cases impose limits to their usage when models have too many parameters. While improvements in GPU hardware might alleviate the need for these requirements, they can still prevail as datasets become bigger and more complex over time.

One approach to deal with this issue is to minimize such costs by using efficient model architectures (see Section 2.2.2). Several model architectures have been proposed over the years that attempt to be efficient so that these costs are managed according to the performance returns. Another approach to solve this issue is to scale the hardware according to the training needs by using distributed computing methods. For example, one could split each batch into

equal parts such that each part is distributed across a different range of GPUs. Another alternative is to use specialized hardware like a Tensor Processing Unit (TPU), which is available through cloud computing platforms like Google Cloud[3]. These types of hardware can achieve major improvements in cost-energy performance when compared to Graphics Processing Units [41].

3.4.5 Workflow Integration

Recent works [5][3][14] demonstrate that deep neural networks can outperform a dermatologist's diagnosis abilities, which makes them an excellent candidate to support clinicians in the decision-making process. However, the research focus of these automated skin lesion diagnosis models does not address the clinical usefulness and practicality of these procedures. Ultimately, the intent of these systems should be on augmenting already established procedures rather than replacing them.

Collaborative working between academic institutions and dermatology centers may prove a useful asset to both sides since the development of expert systems requires multidisciplinary knowledge. Joining forces and sharing expertise are key elements to ensure improvements that have a measurable impact. On the one hand, universities and research institutions may benefit from direct access to the field, the ability to test theories, and the opportunity to apply research in practice. On the other hand, health providers gain from the input of academic expertise for operational problems and technological advances that can be effective in improving health outcomes.

Furthermore, some theorists argue that involving the end-user (medical professionals) in the process of developing and evaluating these systems can ultimately help in the integration of these tools into a production environment [17]. The argument behind this statement is that the feedback received during development is crucial for the adaptation of these systems in the clinical workflow. Additionally, by involving skin professionals in the feedback loop, presumably, the trust of physicians in these types of systems will also increase. Moreover, one can also increase awareness of the requirements related to regulatory, ethical, and legal hurdles during the prototype phase through this feedback [17].

[3] `https://cloud.google.com/tpu`

CHAPTER 4

Experimental setup

This chapter portrays the scope and the environment used for the presented experiments. More specifically, Section 4.1 describes the scope and research design of this work. Section 4.2 presents an overview of the training dataset from the ISIC 2019 challenge, discuss its class distribution, and also outlines the dataset for the unknown class. Section 4.3 explains the preprocessing steps applied to the dataset samples before training. Section 4.4 catalogues several augmentation techniques used further in some of the experiments, along with examples. Section 4.5 describes the dataset split process into train, validation and test sets. Section 4.6 provides a synopsis of the hardware resources used for the experiments. Finally, Section 4.7 presents the software stack used for training and testing models, along with arguments for the choices of the frameworks and tools used.

4.1 Experimental Scope

This work is concerned with the development of a multi-class dermoscopic image classifier for the diagnosis of different types of skin lesions. The classification problem will be addressed using CNN models, focusing on the process of designing the classifier from a set of training examples and the process of evaluating the generalization performance of the classifier.

To create an end-to-end CNN classifier of skin lesions through deep neural networks, one of two approaches can be chosen. The first is to create a model from scratch, which implies designing its architecture, cross-validation a wide range of hyperparameters, choosing the weight initialization method and train it from the ground up. However, this approach is troublesome because it requires thorough reasoning for each of these tasks. In practice, this requires cross-validating a wide range of hyperparameters and network design related parameters, which can be time-consuming and computationally expensive.

However, another approach is using transfer learning by leveraging the weights of pre-trained models. This is a good option when data is scarce, which is often the case for medical imaging-related classification tasks [17]. Additionally, there is a wide range of pre-trained models to choose from, because of benchmark challenges such as the ILSVRC. For the

presented experiments, the choice towards which models to study is based on commonly used ones presented in the literature (see Section 3.3.1), and in their availability through the `tf.keras.applications`[1] framework. More specifically, VGG16, VGG19, ResNet50, ResNet101, ResNet152, InceptionV3, InceptionResNetV2, DenseNet121, DenseNet169, and DenseNet201.

Furthermore, although EfficientNets are not available in the `tf.keras.applications` framework, some of its variants will also be tested due to their remarkable performance on the ImageNet dataset (see Table 2.1). Namely, the shallower and smaller variants such as the EfficientNetB0, the EfficientNetB1, and the EfficientNetB2. Even though there are bigger EfficientNet models, the hardware setup used (further described in Section 4.6) is not able to properly train models larger than EfficientNetB2. Therefore, it has been decided to left them out of this work.

Moreover, the holdout method will be used to make assessments, in which a different set of data to the training set, called the validation set, is used so to obtain an unbiased evaluation 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 [110]. Even though the holdout method has its downsides, the time requirements are much lower compared to techniques like k-fold cross-validation, which is appealing to speed up experiments.

One important challenge in applying CNN to medical imaging is the large amount of labeled data required in order to achieve a performance that surpasses the shallow neural networks. The contributes of the ISIC have been relevant in providing an open-source public access archive of skin images. This publicly available archive has been used for teaching purposes and the development of automated skin lesion diagnosis systems [24]. At the same time, the challenges proposed annually around a specific dataset facilitate better comparisons between new and existing solutions by standardizing evaluation criteria. The difficulties in monitoring and comparing progress in the field are the result of several factors, among which stand out non-standardized evaluation metrics, the use of different datasets, and differences in how learning tasks are framed [15]. Therefore, this work will use the ISIC 2019 challenge dataset which provides 25331 image samples distributed across the 8 categories.

In Section 2.6, several metrics were presented to assess the performance of a machine learning model. Even though there are multiple metrics that one could use for multi-class problems with unbalanced data, the ISIC 2019 organizers argue that BMA is a good metric to compare different approaches on highly unbalanced datasets, because it penalizes strategies that optimize accuracy by preferring predictions in favor of the most prevalent class [14]. Hence, it is beneficial to use the BMA as the primary metric of comparison in the presented experiments.

Furthermore, metrics like accuracy still strongly appear in the literature and can be used to understand whether a model is underperforming on underrepresented classes of the test set. This can be done by looking at the discrepancy between the accuracy and BMA scores on the

[1] `https://www.tensorflow.org/api_docs/python/tf/keras/applications`

test set, in which if the accuracy is superior to the BMA, most likely, underrepresented classes are underperforming in comparison with overrepresented classes. Therefore, the accuracy score will also be used to compare results in some of the experiments.

4.2 Data Representation

4.2.1 Description

The ISIC 2019 challenge [24] provides a training dataset with 25331 samples distributed across 8 different categories that can be represented within a hierarchy shown in Figure 4.1. Such categories are:

- Melanocytic nevus (NV): benign neoplasms (excessive growth of tissue) of melanocytes (cells in the skin that produce and contain melanin) that appear in different variations that can be quite different from each other from a dermatoscopic point of view [92];
- Melanoma (MEL): a malignant neoplasm that is created from melanocytes and may appear in different variants. Melanomas either penetrate beyond the epidermis into the dermis (invasive) or not (in situ). If found the in early stages, Melanoma can be cured by surgical excision (*i.e.*, removing the tissue) [92];
- Dermatofibroma (DF): benign skin lesion caused by an agglomeration of cells within the deeper layers of the skin. It can be caused by an adverse reaction to an injury (*e.g.*, bug bite or splinter) [92];
- Vascular lesion (VASC): Birthmarks or childhood abnormalities of the skin and underlying tissues. It is composed of a collection of samples from cherry angiomas, angiokeratomas, pyogenic granulomas, and hemorrhages [92];
- Benign keratosis (BK): Benign lesions which include samples of solar lentigo (patch of darkened skin), seborrheic keratosis (harmless skin growth), and lichen planus-like keratosis (inflammatory reaction arising from solar lentigo or seborrheic keratosis) [24];
- Basal cell carcinoma (BCC): the most common type of skin cancer [1] that rarely metastasizes but grows destructively if untreated. It may appear in different variants such as flat, nodular, pigmented or cystic [92];
- Squamous cell carcinoma (SCC): the second most common form of skin cancer [1], characterized by abnormal, accelerated growth of squamous cells. When caught early, most SCCs are curable [1];
- Actinic keratosis (AK): also called solar keratosis, it is a kind of abnormal skin cell development as a consequence of genetic code damage. It is considered to be an early form of invasive squamous cell carcinoma [92].

This dataset is an agglomerate of samples from the HAM10000 [92], BCN_20000 [111], and MSK [112], all of which are labeled datasets of skin lesions. The HAM10000 dataset is an effort to boost research on the automated diagnosis of dermatoscopic images that focuses on the quality and reliability of a large volume of data. Images from HAM10000 belong to different sources and go through a pipeline to clean data. All of the samples are manually reviewed by dermatology professionals and non-dermatoscopic images or unreliable diagnoses

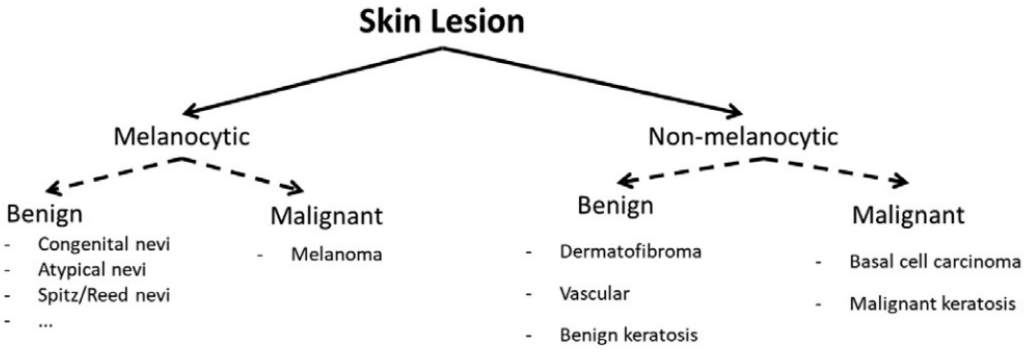

Figure 4.1: Hierarchical organization of skin lesions. Taken from Barata *et al.* [107].

are filtered out. Finally, all of the samples have the same resolution of 600x450 pixels centered around the lesion (see filtering process in Figure 4.2). 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-processing methods before training.

The HAM10000 had already been used in task 3 of the ISIC 2018 challenge. The literature suggests that the top-scoring approaches of the ISIC 2018 surpassed expert dermatologists [14]. However, as Tschandl *et al.* pointed out, the algorithms performed worse on images from other dermoscopic data sources that were not present on the HAM10000 dataset [14]. This statement shows that the HAM10000 dataset alone does not provide enough samples that are representative of real-world scenarios. For example, real-world samples might not have the lesion centered or might be taken under different conditions (*e.g.*, angle, luminosity, or contrast variations).

As a countermeasure, 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 [111], which contains hard to diagnose images of size 1024x1024 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, meaning that some samples belong to the same lesion.

Finally, the last new dataset is the MSK dataset, which contains images of multiple resolutions and multiple aspect ratios. The MSK is composed of multiple smaller datasets, namely:

- MSK-1 which contains images both from benign and malignant melanocytic lesions, but not taken with modern digital cameras. Almost all diagnoses were confirmed by histopathology reports, and the remainder consists of benign lesions confirmed by clinical follow-ups;
- MSK-2 composed of biopsy-confirmed melanocytic and non-melanocytic lesions. This

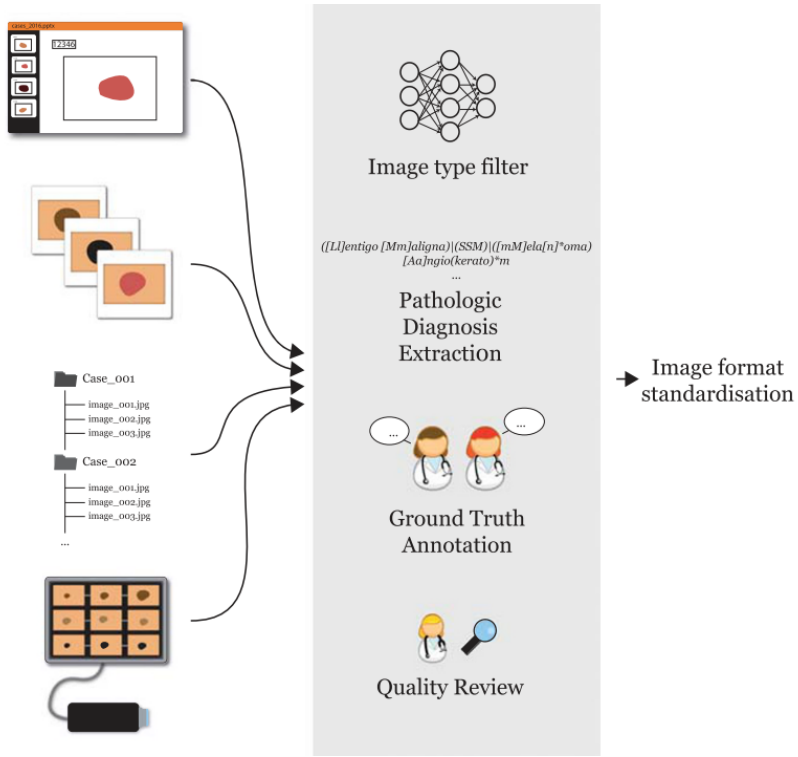

Figure 4.2: HAM10000 sample filtering process. Taken from Tschandl *et al.* [92].

dataset includes over 500 melanomas. Many images have polarized and contact variants;

- MSK-3 contains assorted images. Mostly composed of nevi and BCC lesions. All diagnoses are confirmed through histopathology;
- MSK-4 composed of images belonging to patients with a personal history, clinical diagnosis, or differential diagnosis of melanoma. All diagnoses are confirmed through 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 at random samples from each of the eight classes of the ISIC 2019 challenge dataset in Figure 4.3, one can observe that samples are quite different from each other. They often vary on aspect ratios, luminosity conditions, segmentation, cropping schemes and even the position of the lesion itself. Even within a particular class, an uninstructed person has a hard time pointing out common features, as some of the lesions can be hypopigmented or not be easily segmentable. Overall, the heterogeneity of this dataset aims to create a challenge that provides samples representative of real-world images of lesions, rather than filtered samples that are easy to diagnose [111].

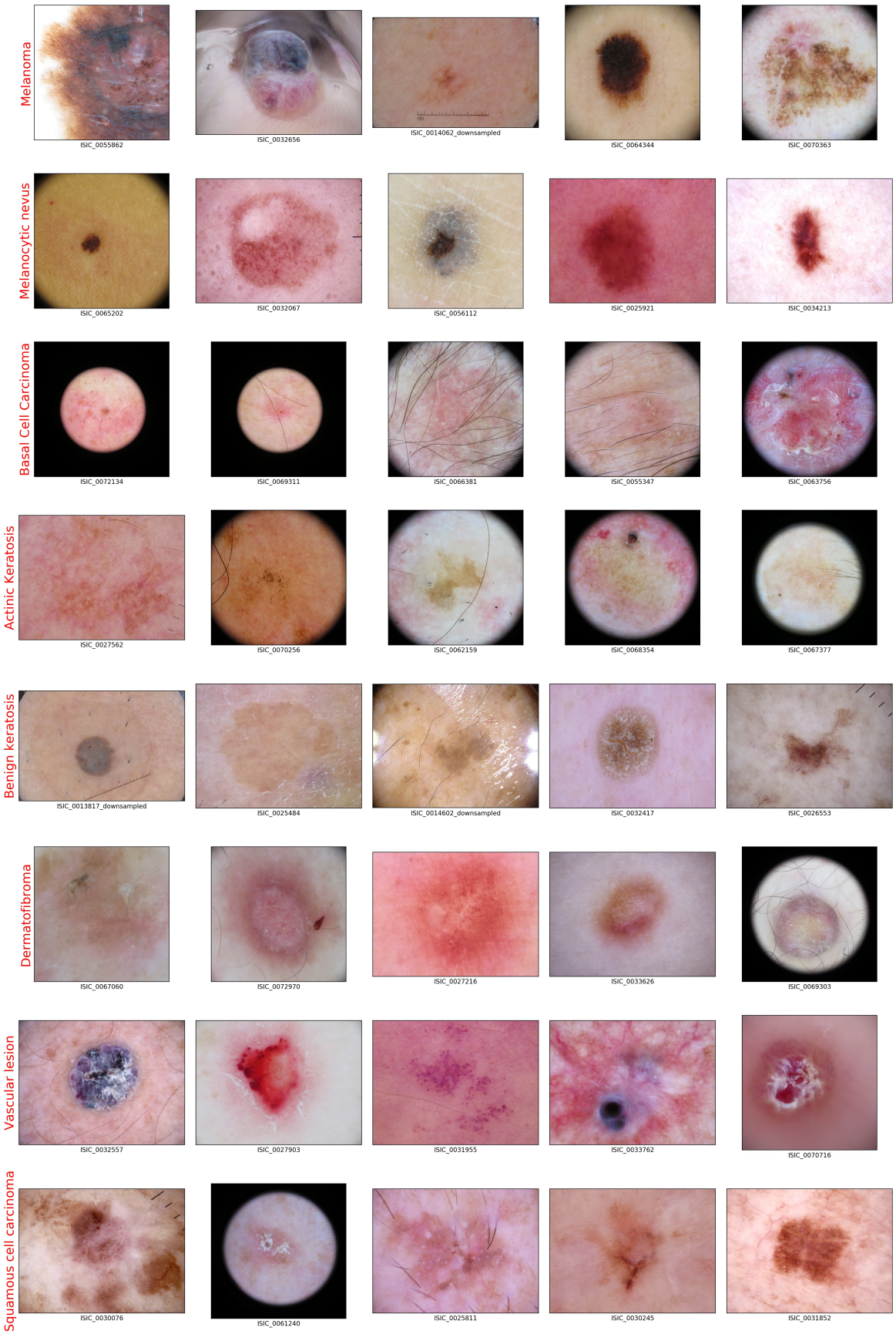

Figure 4.3: Samples from ISIC 2019 training data for the 8 known categories.

4.2.2 Class Imbalance

Note by looking into Figure 4.4 that the dataset provided is highly unbalanced with some classes like the melanocytic nevus (NV) representing almost half of the whole dataset, while a class like dermatofibroma representing less than 1% of the dataset.

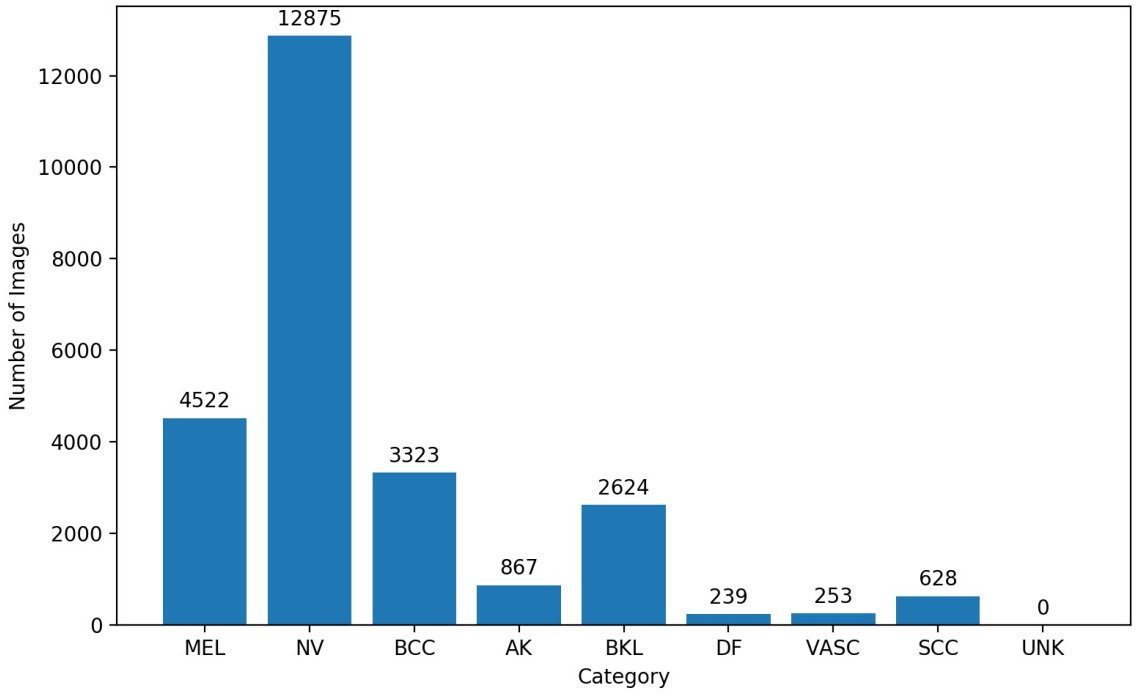

Figure 4.4: Class distribution of ISIC 2019 challenge training dataset.

To tackle this issue, a common method from other ISIC 2019 approaches [21][25][102], is to use some kind of weighted loss function, like the weighted cross-entropy loss function. Following the comparisons done by Gessert *et al.* [20], for the experiments, the weight W_i of each class i is defined as:

$$W_i = \frac{N}{C * n_i} \tag{4.1}$$

where N denotes the total number of samples in the training set, n_i is the number of samples for class i, and C is the number of classes.

This results in the weight distribution illustrated in Figure 4.5. One can observe that 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.

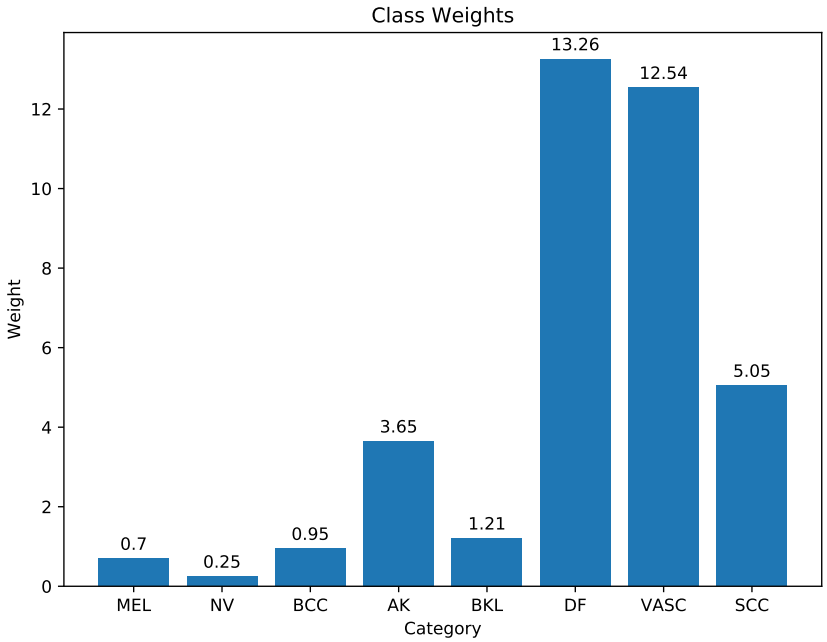

Figure 4.5: Weights for each class of the weighted cross-entropy loss function in the ISIC 2019 training dataset.

4.2.3 Unknown Class

By looking at the sample distribution across the different classes in Figure 4.4 one can observe that no data is provided for the 9th class, namely, the "Unknown" class. This class is meant to represent none of the other classes, and it is meant to address some of the findings of Tschandl *et al.* in which they noted that the accuracy from out-of-distribution test images was considerably worse than in-distribution test images [14].

The range of skin lesion diagnosis extends beyond the 8 classes represented in the ISIC 2019 dataset (see Figure 4.4). 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 part of the original training distribution.

Different strategies to deal with this unknown class will be experimented with in Chapter 6. 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 4.1 shows the distribution of such samples.

All of the samples come from the ISIC archive [93] and are left out from the ISIC 2019 challenge because they do not belong to any of the original 8 classes. More specifically, they are part of the MSK dataset, which means that samples follow the same conditions to the other samples from the same dataset. This makes them quite similar to images from the ISIC 2019 challenge training dataset for an uneducated observer (see Figure 4.6). As such, these samples will receive the same pre-processing as the samples from the original training dataset.

Lesion category	Samples amount	Source dataset
Angiofibroma or fibrous papule	8	ISIC Archive
Scar	4	
Angioma	12	
Atypical melanocytic proliferation	12	
Lentigo simplex	22	
Lentigo NOS	70	

Table 4.1: Samples per category of skin lesion, for the "unknown" class.

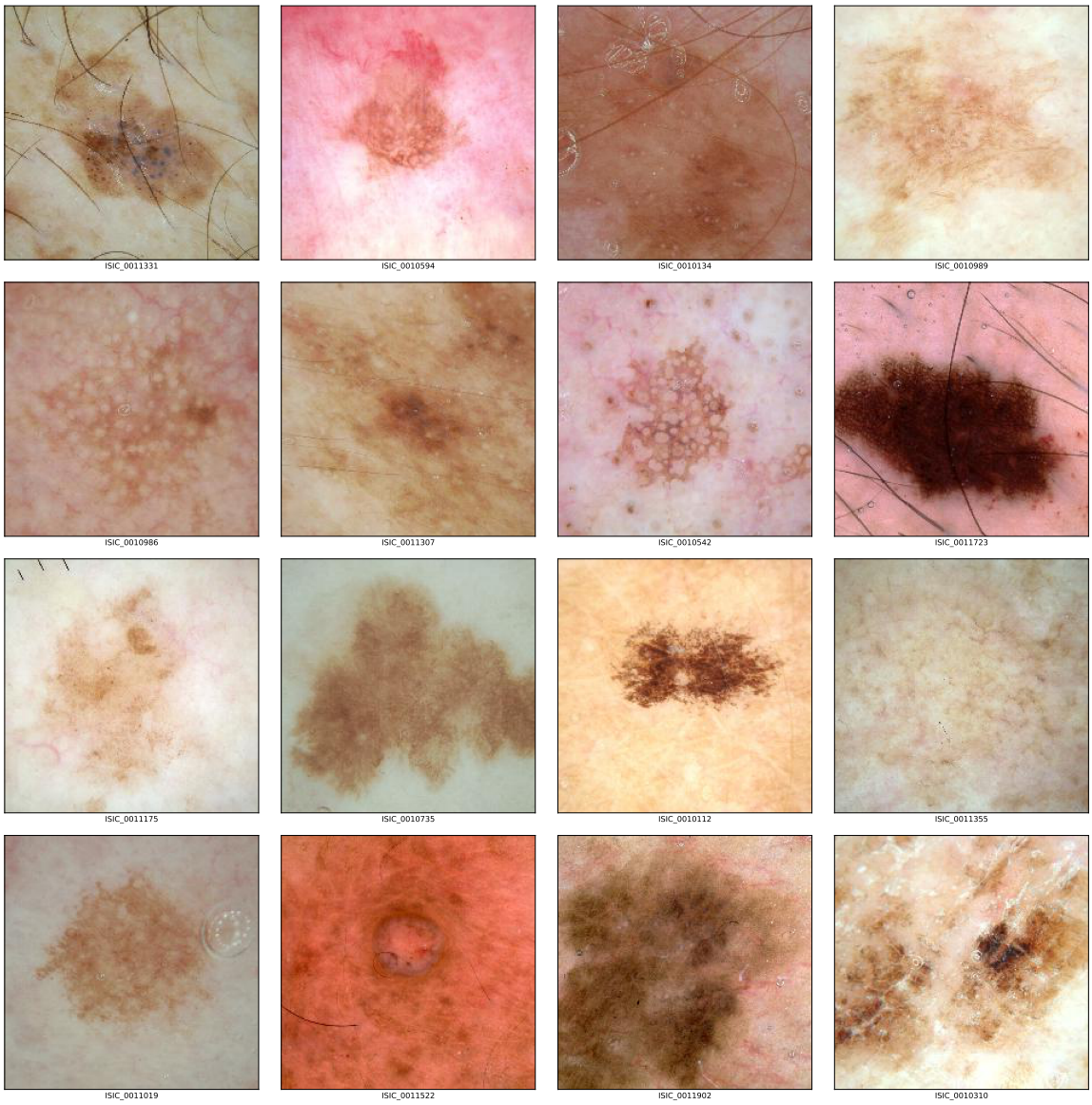

Figure 4.6: Randomly chosen samples from the dataset created for the "unknown" class.

4.3 Data Preprocessing

Each sample of the dataset undergoes several preprocessing steps, more specifically:

1. Most readily available pre-trained 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, naively resizing them to the network's input tensor dimensions without regard to each 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):
    """
    Center crops the image with a squared aspect ratio (1:1).

    # Arguments
        img: PIL image.
    # Returns:
        the cropped image.
    """
    width, height = img.size
    # If already a square do nothing
    if width == height:
        return img

    length = min(width, height)

    # Define center crop box
    left = (width - length) // 2
    upper = (height - length) // 2
    right = left + length
    lower = upper + length

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

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

2. Even though different model architectures use different input tensor sizes, all images should be resized to the target network's input dimensions as soon as possible in the data pipeline. One can argue that by doing this process earlier will reduce the computational costs of applying this operation during the training process. Therefore, images are resized using nearest-neighbor interpolation before training the models.
3. Based on Gessert *et al.* [20], each image is normalized by subtracting the channel-wise mean of the entire ISIC 2019 training dataset, and then dividing it by its standard deviation (as in code snippet 2). The channel-wise mean and standard deviation of the ISIC 2019 training dataset was calculated beforehand.

```python
def preprocess_input(x):
    """
    Preprocesses a numpy array encoding a batch of images.
    Each image is normalized by subtracting the mean and dividing by the standard deviation channel-wise.

    # Arguments
        x: a 3D numpy array consisting of RGB values within [0, 255].
    # Returns
        Preprocessed array.
    """
    if not issubclass(x.dtype.type, np.floating):
        x = x.astype(K.floatx(), copy=False)

    # Convert pixel values from [0, 255] to [0, 1]
    x /= 255.

    # Mean and standard deviation calculated over the entire ISIC 2019 training dataset
    mean = [0.6236, 0.5198, 0.5038]
    std = [0.2422, 0.2235, 0.2315]

    # Zero-center by mean pixel
    np.mean(x, axis=(0, 1))
    np.std(x, axis=(0, 1))

    # Subtract mean from pixel values in the 3 RGB channels
    x[..., 0] -= mean[0]
    x[..., 1] -= mean[1]
    x[..., 2] -= mean[2]

    # Divide pixel values by standard deviation for all 3 RGB channels
    if std is not None:
        x[..., 0] /= std[0]
        x[..., 1] /= std[1]
        x[..., 2] /= std[2]
    return x
```

Code Snippet 2: Function that normalizes the sample images over the entire ISIC 2019 dataset

4.4 Data Augmentation

Further in Chapter 5 and Chapter 6, two types of data augmentation are going to be used, namely:

- Offline data augmentation used as a method to oversample underrepresented classes. More specifically, it is going to be used to class balance samples.
- Online data augmentation used as a method to reduce overfitting during training.

However, in both of these methods the same rules are used, more specifically, each synthetic sample is generated from a randomly selected sample from a specific class, and is generated by an array of image processing techniques each with a probability of 0.5 of being used. Therefore, the selected sample, which will serve as a basis for augmentation, will likely be different each time. Additionally, if a sample is selected more than once, there is a high chance of generating a different sample as the augmentations performed will likely differ. By generating a different sample each time augmentation is performed, the risk of the model being trained with the same samples over and over again, possibly leading to overfitting, is minimized.

The augmentations used take advantage of the `Augmentor`[2] library, which provides a wide range of data augmentation techniques, with specific implementations for the `tf.keras` framework. The following are the ones used throughout some of the experiments presented in Chapter 5 and Chapter 6:

- Horizontal flips;
- Vertical flips;
- 90º rotations;
- Randomly increase or decrease contrast;
- Randomly increase or decrease brightness;
- Randomly increase or decrease color intensity;
- Randomly erase a small section of the image;
- Shears on the x axis or y axis with 20 degrees to the left/right or top/down, respectively (see Figure 4.7);

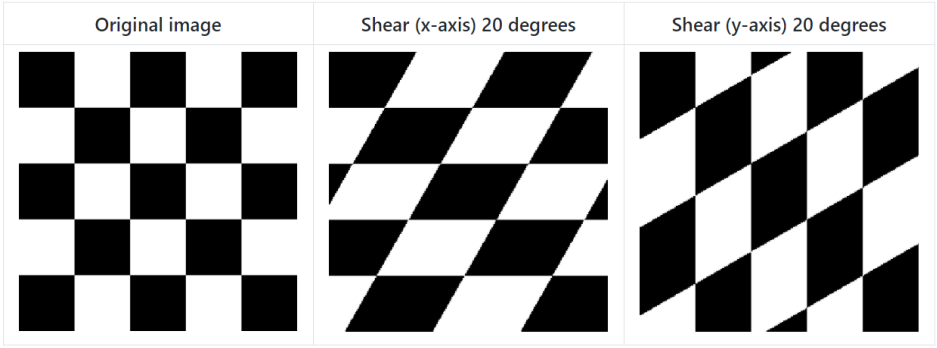

Figure 4.7: Examples of shear augmentations in the x-axis (middle) and y-axis (right). Taken from the Augmentor[3].

- Tilts forward, backward, left, or right (see examples in Figure 4.8);

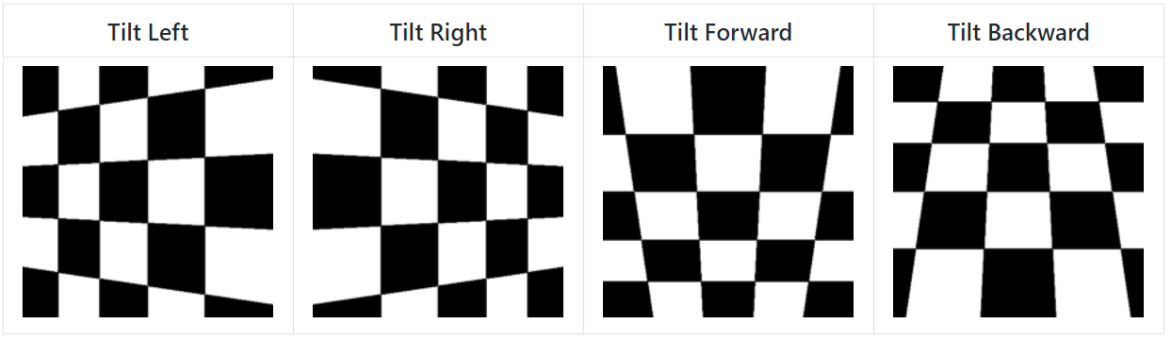

Figure 4.8: Examples of tilt augmentations from forward the x-axis (middle) and y-axis (right). Taken from the Augmentor library.

- Skew image towards a random corner of the image on the x-axis or y-axis (see examples in Figure 4.9);
- Distort the original image (see example in Figure 4.10).

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

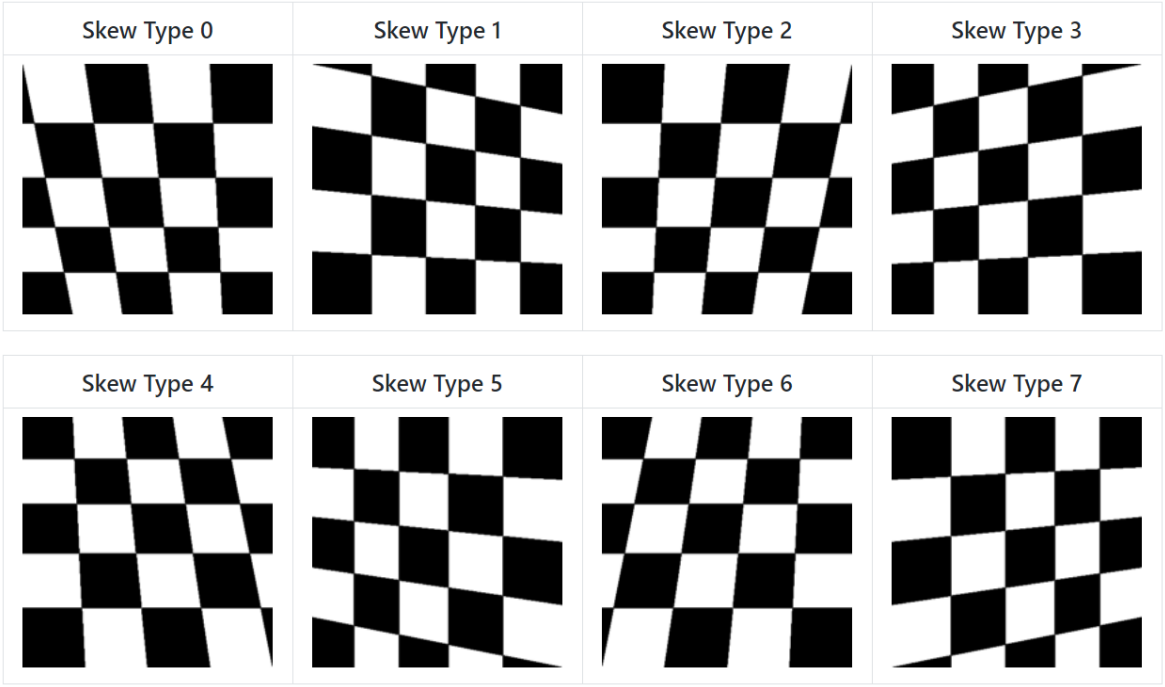

Figure 4.9: Examples of skew augmentations towards different corners of the image either on the x- or y-axis. Taken from the Augmentor library.

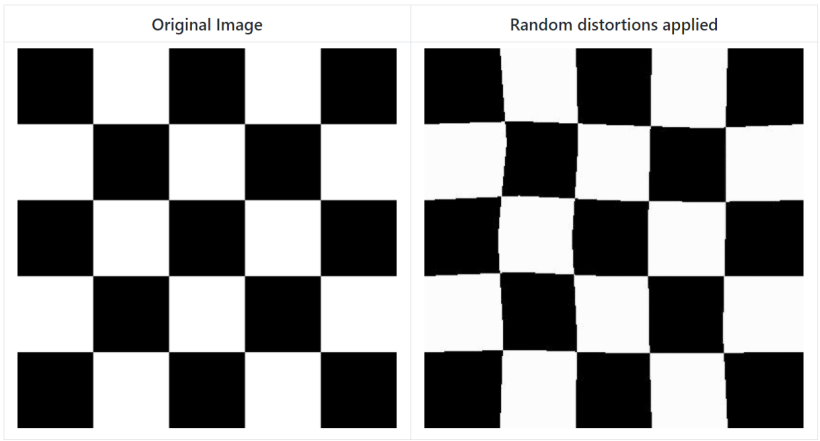

Figure 4.10: Example of an distortion applied as an augmentation method. Taken from the Augmentor library.

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 4.11 illustrates the examples of these augmentation techniques being performed to an original sample. 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 [113] were also considered, but one could argue that such techniques do not make sense in skin lesion classification. These type of

[3] `https://github.com/mdbloice/Augmentor`

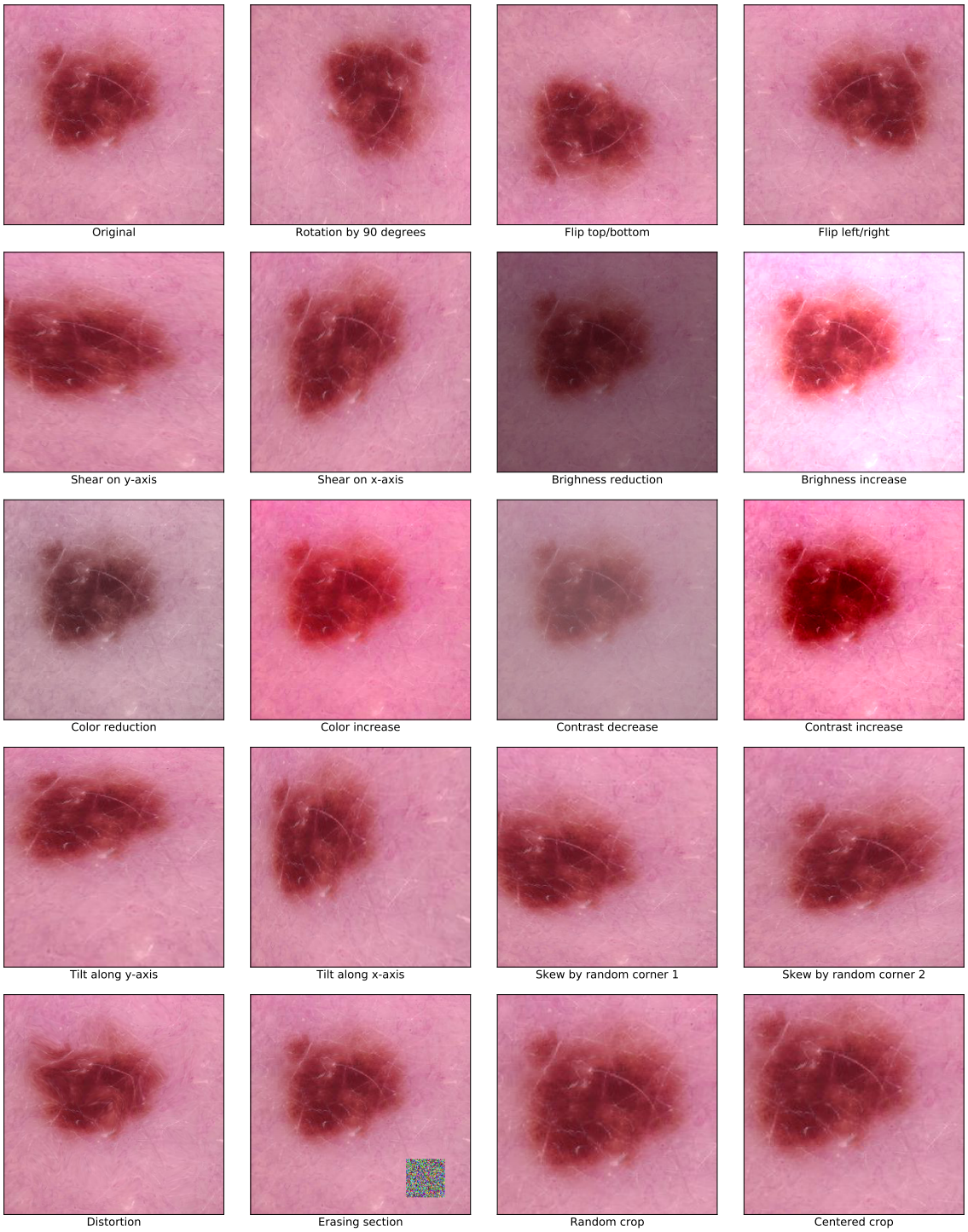

Figure 4.11: Examples of image processing augmentation techniques used in Chapter 6.

augmentations 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.

4.5 Data Split

The original training and test datasets from the ISIC 2019 challenge are available for direct download. However, the test set is not labeled because the ground-truth information is used internally by the organization for computing performance metrics and rank submissions. As such, for this study, samples from the ISIC 2019 challenge training dataset will be split into training, validation, and test sets.

A fixed validation scheme will be used instead of a cross-validation one, in order to minimize the computational cost of the experiments. To compensate for this lack of averaging over multiple folds of the data (which gives statistical confidence in the results), the same validation and test sets are used across different experiments by using the same random initial state (see code snippet 3). In practice, this means that parameters are initialized identically between experiments which provides some level of statistical confidence when making comparisons and guarantees reproducibility of the results.

```
def train_test_split(df, test_size=0.1):
    """
        Split ensures reproducibility as the random state (seed) is always the same.
        Returns the split of dataframe df in a stratified manner.

        # Arguments
            df: dataframe containing the dataset samples where the "category" column
                indicates the true lesion category
            test_size: size of the test set
        # Returns
            train and test datasets.
    """
    return train_test_split(df, stratify=df['category'], test_size=test_size, random_state=42)
```

Code Snippet 3: Split function that is used to split the ISIC 2019 training data into train, validation and test sets. It also ensures that the train, test and validation sets are always the same between different runs.

Moreover for this process, the ISIC 2019 training dataset is split into train and test sets in a 90%-10% stratified fashion, which means that 2517 samples from the original set will be part of the test set. The remaining 90% will be split again using a 90%-10% stratified split for the train and validation sets, respectively, which means that approximately 20518 samples will be used for train and 2280 for validation. However, for out of training distribution detection related experiments, more 128 samples will be used for the "unknown" class, which are also split in the same way as the original 8 classes.

Overall, the distribution of samples from different classes into train, validation, and test sets can be seen in Figure 4.12. One can observe that like the original ISIC 2019 training dataset, all three train, validation, and test sets are highly imbalanced, with classes like the melanocytic nevus (NV) representing almost half of these sets.

Although one could use a larger amount of samples for the validation and test sets, that would also mean that fewer samples would be used for the training dataset. As such, a compromise was taken in order to train the model with more samples, even though the test

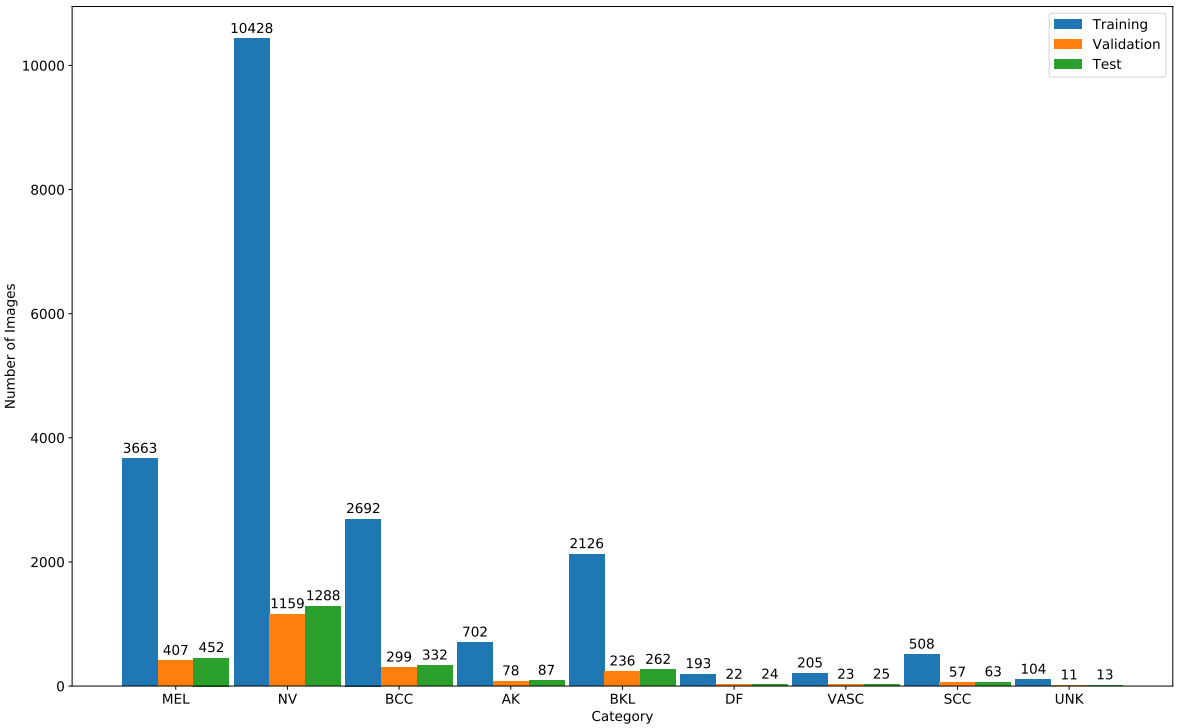

Figure 4.12: Sample distribution used for the train, validation and test sets across 9 different classes.

set results might contain some bias associated with the low amount of samples. Other authors like Bissoto *et al.* [98] from ISIC 2018 also employ a similar split in order to achieve better generalization performance.

As a measure to reduce the impact of class imbalance within the training data, the training set will be oversampled or undersampled in some of the experiments in Chapter 6. However, this process will not be applied to the validation and test sets. An argument can be made that by splitting train and test/validation data after the augmentation is performed, one is creating a high bias in the test/validation data. This is the case because some samples from the test/validation set would be variations of samples from the training set and vice versa. As such, both splits are performed beforehand from non-augmented samples, as a way to attain unbiased performance metrics on the test and validation sets. Moreover, the test and validation sets suffer the same preprocessing steps from the training set (central crop, resizing, and image normalization).

4.6 Hardware

The presented training tasks require high computational resources, specially, in terms of memory and graphic processing power, making it a requirement to have a state-of-the-art computer in order to properly train and test deep learning models. As such, a request has been made to the Laboratory for Automation and Robotics (LAR) team at Department of Mechanical Engineering at the University of Aveiro to access their deep learning research server codenamed Deeplar (see Figure 4.13).

Figure 4.13: Deeplar, the computer used for the experiments of the presented research work.

It has four state-of-the-art GPUs 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.

4.7 Software

Deeplar runs on a distribution of Linux called openSUSE Tumbleweed 20191004[4]. 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 [5]. 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. Specifically, Deeplar uses version 7.6.0 [6].

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[7], due to its 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[8] and took advantage of the following packages:

- TensorFlow[9] 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, which abstracts most of the logic behind a simple framework;

[4] `https://software.opensuse.org/distributions/tumbleweed`
[5] `https://developer.nvidia.com/cuda-zone`
[6] `https://developer.nvidia.com/cudnn`
[7] `https://docs.conda.io/en/latest/miniconda.html`
[8] `https://www.python.org/`
[9] `https://www.tensorflow.org/`

- NumPy[10] 1.15.4 is used for various vector and matrix operations which eases the development process;
- Pandas[11] 1.0.1 was chosen to analyze and manipulate large amounts of structured data;
- Pillow[12] 5.4.1 for image handling and transformations because of this work's image preprocessing needs;
- scikit-learn[13] 0.20.2 for calculating multiple metrics and to split data into train/validation and test sets;
- jupyter[14] 1.0.0 was used as convenience tool for analysing results and creating graphs in a interactive way;
- matplotlib[15] for visualizing results through a multitude of graphs.

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

- Augmentor[16] 0.2.8. Used as an image augmentation library that provides a wide range of simple and complex augmentation operations. This allowed the focus to be on the optimization of the models rather than the implementation of these image processing augmentation algorithms;
- EfficientNet Keras[17] 1.15.4. This is an open-source implementation of EfficientNets for the Keras framework. TensorFlow 2.0.0 does not provide an implementation of EfficientNet, however, it is scheduled for future releases to be integrated within the tf.keras framework.
- Keras implementation of ODIN[18] [67]. The ODIN implementation is featured alongside the source code for the Hsin-Wei Wang's ISIC 2019 approach and mimics the original pyTorch implementation done by Liang *et al.*[19] (a team of Facebook researchers).

[10] https://numpy.org/
[11] https://pandas.pydata.org/
[12] https://pillow.readthedocs.io/en/stable/
[13] https://scikit-learn.org/
[14] https://jupyter.org/
[15] https://matplotlib.org/
[16] https://github.com/mdbloice/Augmentor
[17] https://github.com/qubvel/efficientnet
[18] https://github.com/wanghsinwei/isic-2019
[19] https://github.com/facebookresearch/odin

CHAPTER 5

Pre-trained Model Choice and Parameter Optimization

In Chapter 3, it was recognized that deep learning in the context of skin lesion diagnosis is a rapidly developing area that ultimately can provide several benefits for both dermatologists and patients. However, to create a deep learning based skin lesion classifier, one must first address some concerns related to the choice of the pre-trained model, and the optimization of hyperparameters.

This chapter provides results related to these optimization concerns in a systematic way, such that comparisons can be made and meaningful conclusions can be drawn. First, Section 5.1 studies which pre-trained models work well for skin lesion classification and the overall impact of different transfer learning approaches in this domain. Next, Section 5.2 methodically optimizes the chosen model's hyperparameters through common reasoning to improve the overall performance of the model. Finally, Section 5.3 discusses and compares the obtained results.

5.1 Pre-trained Model Choice

In Chapter 2, several CNN models from different architectures (*e.g.*, ResNets, DenseNets, VGGNets, and EfficientNets) pre-trained on the ImageNet dataset [18] were presented. These models are often scaled versions of baselines which allows a particular architecture to have a wide range of models with different properties (*e.g.*, depth, input resolution or number of trainable parameters). However, one can argue that the weights optimized towards the ImageNet dataset do not translate well into skin lesion classification because this dataset contains far different classes (*e.g.*, dogs, cats, birds) from skin lesion dataset's classes (*e.g.* melanoma, dermatofibroma). As such, in order to re-purpose a pre-trained model towards skin lesion classification one must use transfer learning (see Section 2.3).

However, one must first filter which pre-trained models should be considered for the problem of skin lesion classification. More specifically, which models from the VGG, ResNet,

DenseNet, Inception, and EfficientNet pre-trained on ImageNet work well for the problem of skin lesion classification and in what conditions. In order to answer these open questions, experiments will be made under the following settings to provide a fair evaluation for each of these models:

- Each model is trained on an undersampled version of the ISIC 2019 dataset with 5000 training samples, that maintains the original class distribution. The undersampling process is done by randomly seleting samples from the original dataset in a stratified manner. This smaller dataset will substantially decrease each model's train time, which will allow us to train more pre-trained models in an agile manner. An undersampled version of the original dataset will likely yield similar conclusions as if the whole dataset was being used;
- Online data augmentation is performed with random crops, flips, and rotations each with probability 0.5 to reduce overfitting while training;
- An global average pooling layer is introduced at the end of the convolutional base in order to reduce the number of parameters for the classifier;
- The original pre-trained model's classifier is replaced by a new classifier composed of one fully-connected layer with 512 neurons which use 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 highest softmax probability class. In Figure 5.1 one can see 3 examples of samples taken from ISIC 2019 dataset being classified into probabilities by one of the models (DenseNet201);
- The classifier weights are initialized using Glorot's initialization [34];
- Following the work done by Gessert *et al.* [20] the Adam optimizer [33] is used, with $\rho_1 = 0.9$ and $\rho_2 = 0.999$
- For the experiments, a batch size of 32 and a learning rate of 10^{-4} is used. However, the learning rate is reduced by a factor of 10 when validation loss stops improving for 8 epochs. Each model trains for a maximum of 100 epochs but early stopping is performed whenever validation loss stops improving for 16 epochs;
- For each epoch, all the samples are shuffled before being fed into the network;
- For each training process, 3 models are saved: The model that obtained the highest BMA on the validation set, the model with the lowest loss on the validation set, and finally the resulting model from the last epoch. However, the performance on the test set is evaluated according to the model which attained the highest BMA on the validation set.

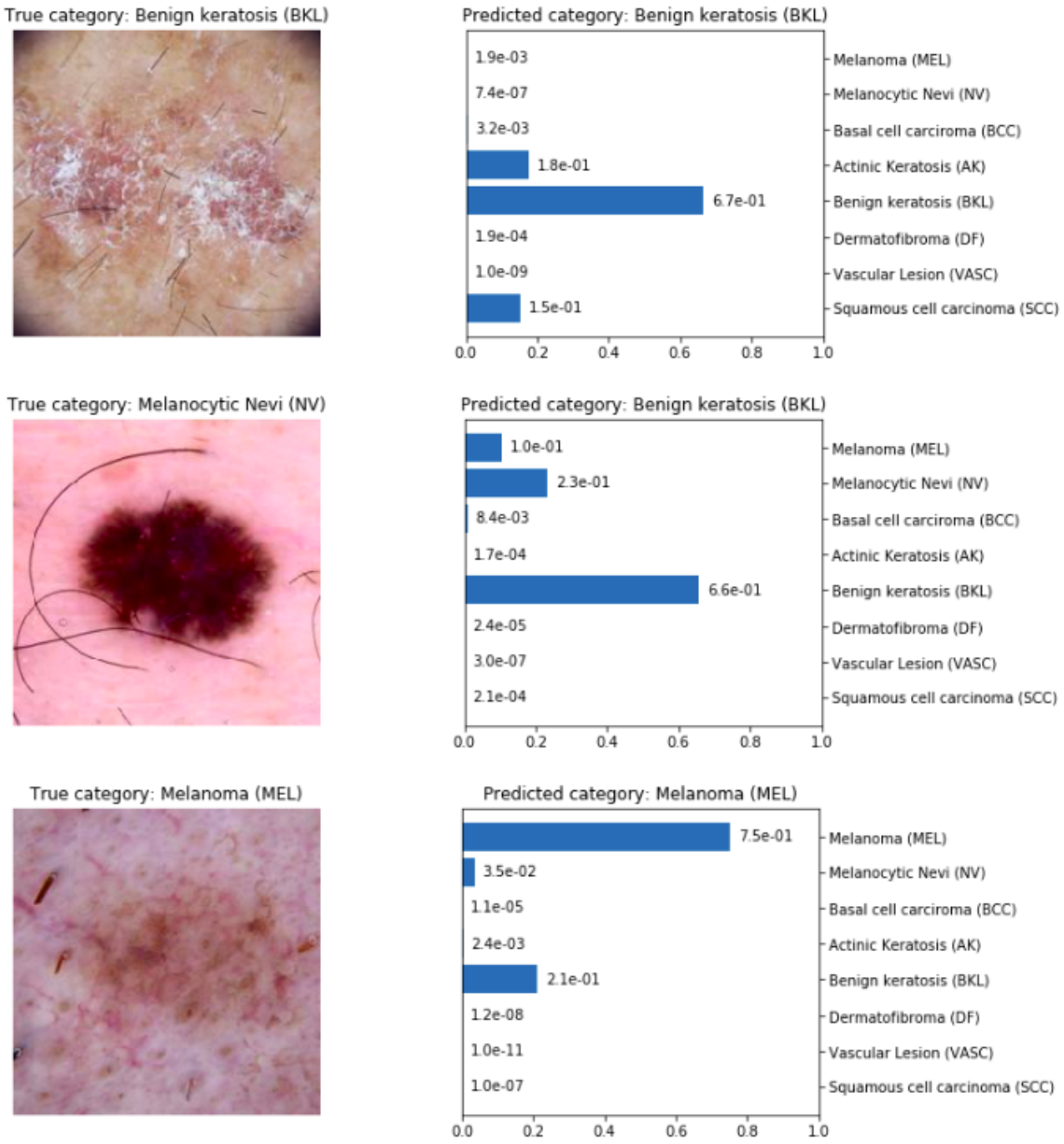

Figure 5.1: 3 Samples of the test set along with their respective softmax probabilities. Results were inferred using the fine-tuned DenseNet201 model.

5.1.1 Freeze the Convolutional Layers

One way of re-purposing a pre-trained model is by simply replacing its 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 (also called the convolutional base). Additionally, the weights from the layers of the convolutional base must be extracted and frozen, which means that they will be transferred directly from their original model and remain unchanged for the duration of the training process. Figure 5.2 compares the accuracy and BMA of applying this approach to several pre-trained models over the test set. One can observe that the overall performance is not optimal for any of the

models, with the best performing architectures being VGG and EfficientNet. These results are to be expected because the weights of the layers in the convolutional base of these models were all trained in a very different dataset (ImageNet), from the ISIC 2019 challenge dataset.

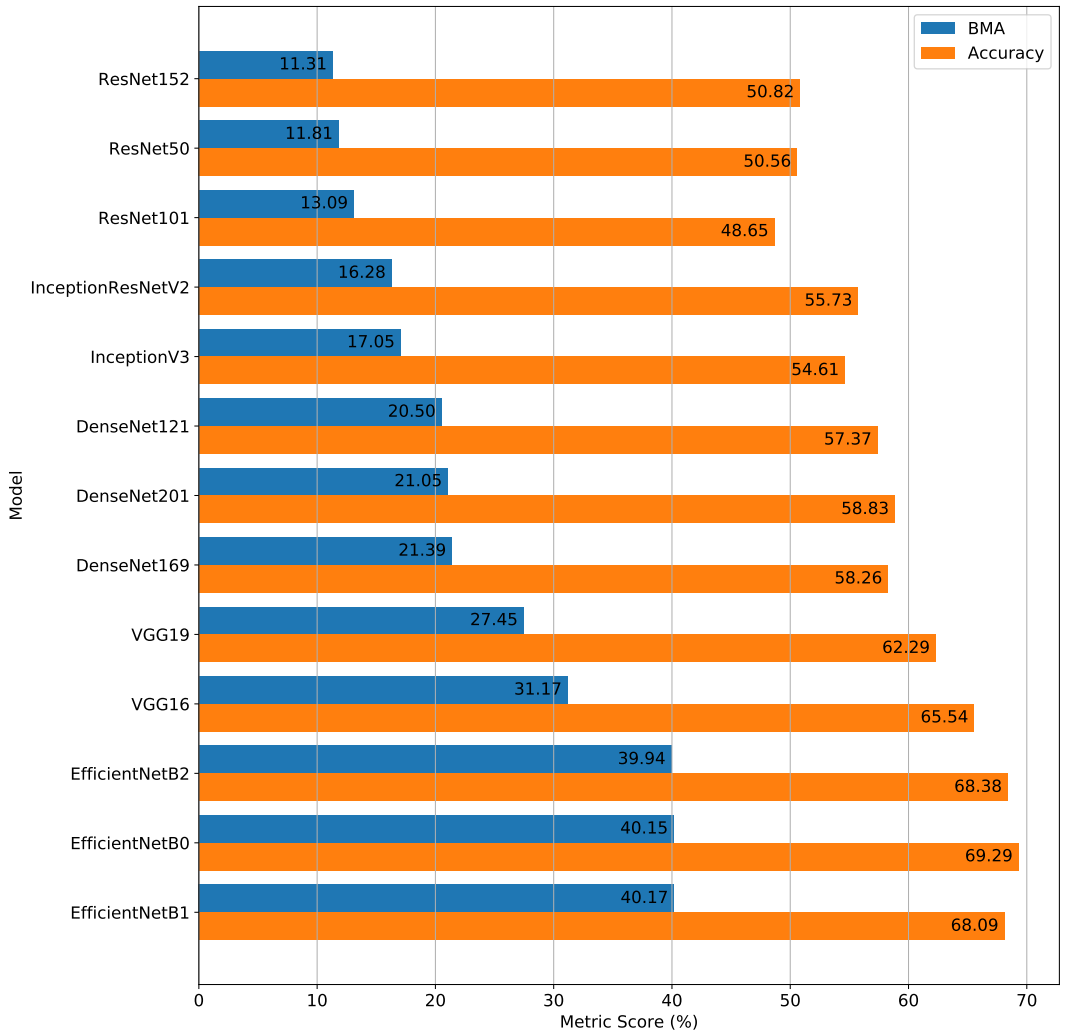

Figure 5.2: BMA and Accuracy on the test set of multiple models pre-trained on ImageNet using transfer learning. Weights from convolutional layers are extracted from their initial model and frozen for the entirety of the training process. Classifier's layers are trained from scratch.

Moreover, it seems like some model architectures are more capable of learning skin lesion related knowledge from only training the classifier. Namely, the EfficientNet family of models attains much better BMA and accuracy scores in comparison with architectures like the ResNet or DenseNet. Presumably, the convolutional base of EfficientNets learns more generalizable knowledge in comparison with other architectures, which makes them more capable of achieving better performance using this transfer learning approach.

5.1.2 Extraction and Fine-tuning of Convolutional Layers

Another approach 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 (weight extraction), but are also updated along with the classifier on each epoch (unfrozen). 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 5.3 shows that this assumption is true, as all pre-trained model achieve a significant increase in both accuracy and BMA when compared with Figure 5.2.

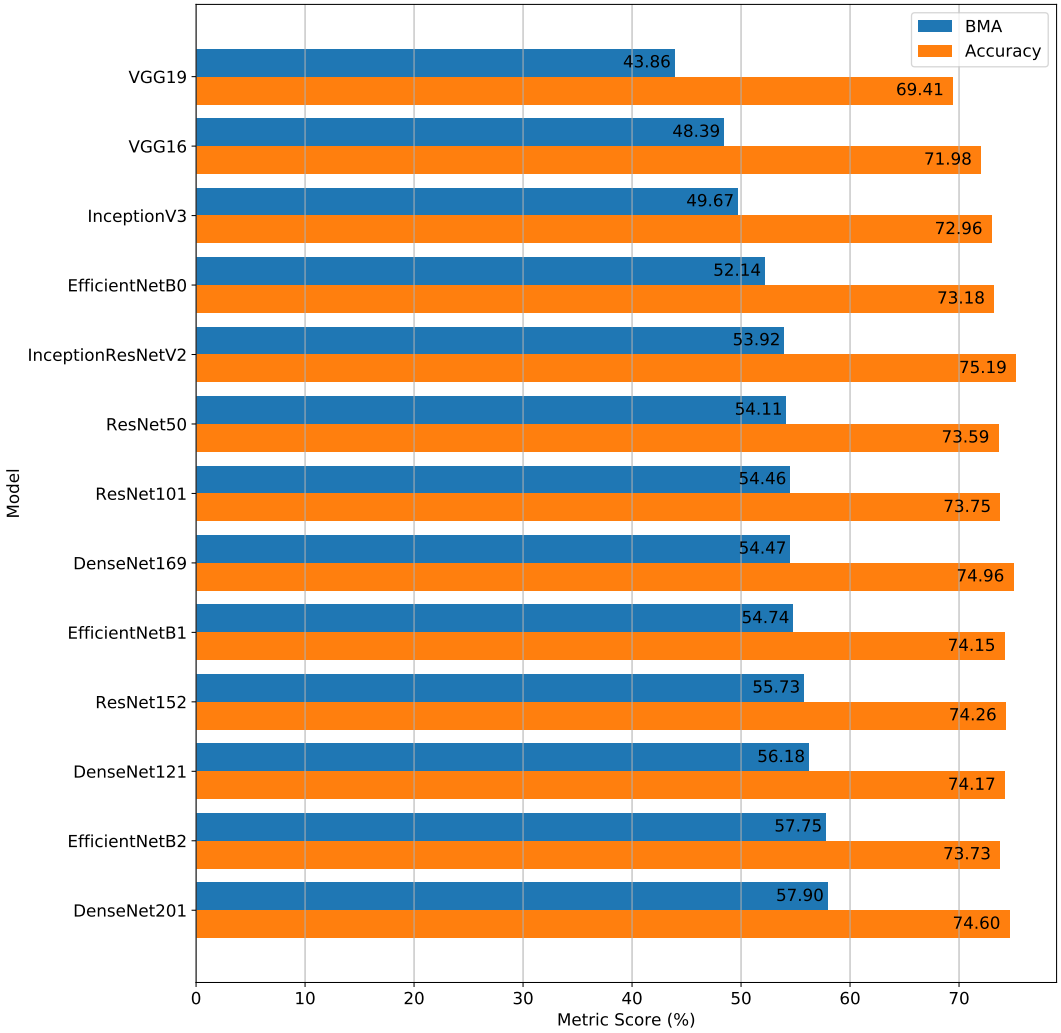

Figure 5.3: BMA and Accuracy on the test set of multiple models pre-trained on ImageNet using transfer learning. Weights from convolutional layers are extracted from their initial model and fine-tuned for the entirety of the training process. Classifier's layers are trained from scratch.

Generally by looking into Figure 5.3, one can observe that more recent architectures such as DenseNet or EfficientNet outperform older architectures such as VGG (see Table 2.1). 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 can create more levels of abstraction. The one exception to the depth rule is presented by the VGG family of models, which does not improve performance with depth increase (*i.e.*, VGG16 and VGG19), presumably, related to the vanishing gradients

problem. This is an issue that happens during the backpropagation of gradients while training, in which the gradient decreases exponentially as it gets propagated down to the initial layers [28]. When undealt, this issue becomes a larger concern for deeper networks like the VGG19.

This problem was later addressed by architectures such as ResNet with the introduction of the skip connection (see Section 2.2.2), which would allow it to create deeper models. Indeed, by looking at Figure 5.3, ResNet pre-trained models outperform older and shallower pre-trained models such as the VGG16 and VGG19. Even within the same architecture, ResNet takes advantage of the depth to further increase performance. DenseNet extended the skip connection concept which allowed it to build even deeper models such as the DenseNet201 with 201 layers. One can observe that the DenseNet pre-trained models outperform most of ResNet, VGG, and Inception pre-trained models, with the DenseNet201 being the best performing pre-trained model.

From looking at Figure 5.3, EffcientNets scale exceptionally well with the amount of trainable parameters available. For example, EfficientNetB0, which is the model with less trainable parameters from all the tested models, has similar performance to the Inception-ResNetV2 which has a much larger amount of trainable parameters (see Table 2.1). In contrast, EfficienNetB2 has similar test set performance to the DenseNet201, while having a considerably lower amount of trainable parameters. One can hypothesize that the scalability of this family of models concerning the number of trainable parameters is related to the compound scaling method presented by Tan *et al.* [44] (see Section 2.2.2). Moreover, one can assume that bigger models such as the EfficientNetB3 or even the EfficientNetB7 could outperform DenseNet201. However, it has been decided not to train and test such models, because of the limited compute capability of Deeplar (see Section 4.6).

Furthermore, one can observe that for all the pre-trained models, the BMA is much lower than the accuracy. This is a consequence of the train and test sets being imbalanced, which makes the model correctly classify more samples towards classes with more data (illustrated in Figure 5.4). The accuracy metric is very sensitive to this type of imbalance, as it does not give the same weight to each class's performance. For the accuracy metric, the weight of each class is proportional to the number of samples of that class on the test set, which makes it very prone to increase or decrease according to the performance of overrepresented classes. In contrast, for the BMA score, the performance of the model in each class is weighted the same (see section 2.6), which substantially lowers the score because underrepresented classes have significantly worse performance than overrepresented classes.

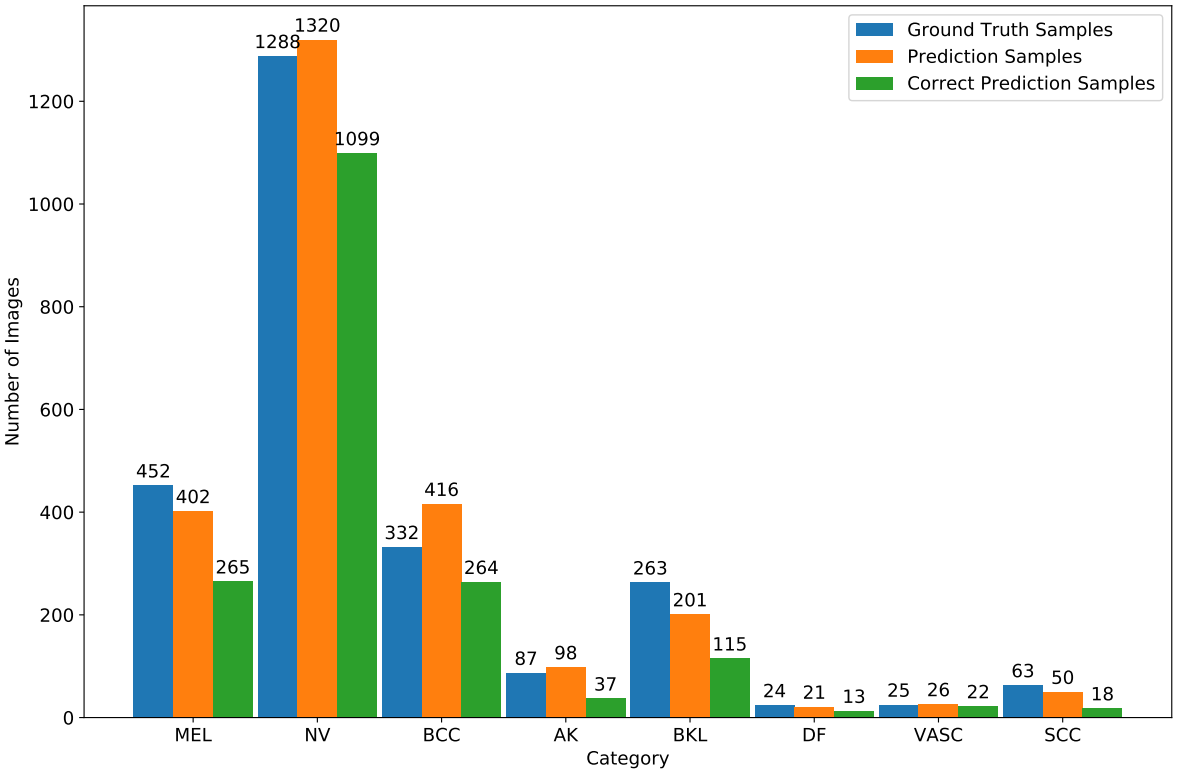

Figure 5.4: Ground truth samples, prediction samples, and correct prediction samples across different classes in the test-set using the fine-tuned DenseNet201 model.

5.2 Hyperparameter Optimization

So far, models were trained with commonly used hyperparameters from the approaches for the ISIC 2019 presented in Chapter 3. However, as a means to optimize these values, one should carefully analyze and test different hyperparameters values, to hopefully improve the test set performance. Therefore, in this section, results from a multitude of experiments with different hyperparameters will be presented.

In order to systematically optimize each hyperparameter without leading to a combinatorial explosion, the following strategy is employed:

- For each hyperparameter, a range of common values from the literature are tested and compared with one another using train and validation BMA;
- The choice of the hyperparameter is made according to an analysis of the results, and all subsequent experiments with other hyperparameters will use that choice;
- This process will be repeated until all hyperparameters are chosen.

Taking this into consideration, a benchmark for each experiment will be set to fairly compare different hyperparameters and to take advantage of the already acquired knowledge from Section 5.1. More specifically:

- Each model is trained on an undersampled version of the ISIC 2019 dataset with 5000 training samples, that maintains the original class distribution. The undersampling process is done by randomly selecting samples from the original dataset in a stratified manner. This smaller dataset will substantially decrease each model train time, which

will allow us to experiment with more hyperparameters in an agile way. An undersampled version of the original dataset will likely yield similar conclusions as if the whole dataset was being used;

- Online data augmentation is performed with crops, flips, and rotations to reduce overfitting while training;
- For the transfer learning approach, all the layer's parameters from the convolutional base of the pre-trained model are extracted and fine-tuned, which yields the best performance for this specific dataset (see the results presented in Section 5.1);
- An global average pooling layer is introduced at the end of the convolutional base in order to reduce the number of parameters for the classifier;
- The original pre-trained model's classifier is replaced by a new classifier composed of one fully-connected layer with 512 neurons which use the ReLU activation function, and one softmax layer with 8 neurons to translate each of the class's probabilities;
- The classifier weights are initialized using Glorot's initialization [34];
- Following the work done by Gessert *et al.* [20] the Adam optimizer [33] is used, with $\rho_1 = 0.9$ and $\rho_2 = 0.999$;
- Each model trains for a maximum of 100 epochs but early stopping is performed whenever validation loss stops improving for 16 epochs, as a measure to reduce overfitting;
- For each epoch, all the samples are shuffled before being feed into the network;
- For each training process, 3 models are saved: The model that obtained the highest BMA on the validation set, the model with the lowest loss on the validation set, and finally the resulting model from the last epoch. However, performance is always evaluated according to the model which attained the highest BMA on the validation set.

5.2.1 Varying the Batch Size

The batch size defines the number of training samples passed through the network in one forward and backward pass. The batch size is directly correlated to the number of iterations per epoch using the Equation 5.1.

$$\text{iterations per epoch} = \frac{\text{number of samples}}{\text{batch size}} \tag{5.1}$$

where one iteration is one backward and forward pass. This means that larger batch sizes will have less iterations per epoch, which generally makes them faster to train. However, larger batch sizes also increase the GPU's memory requirements.

Tests have been made for 4 different batch sizes, more specifically, 4, 8, 16, and 32. Larger batches than 32 are impractical for this experimental setup because Deeplar at LAR (see Section 4.6), runs out of memory on some pre-trained models with large batch sizes(*e.g.*, InceptionResNetV2, DenseNet201).

From the results in Figure 5.5, one can observe that smaller batch sizes improve the BMA scores on the validation set while also having a regularization effect, because train and validation BMA scores are much closer together for smaller batch sizes (*e.g* 4) in comparison with larger ones (*e.g.*, 32). However, one should also consider the disadvantages of using such

small batch sizes, namely, longer times to train the models, which can become troublesome with larger datasets such as the ones used further in Section 6.1. Therefore, as a compromise between agility and performance, a batch size of 8 was chosen for future experiments.

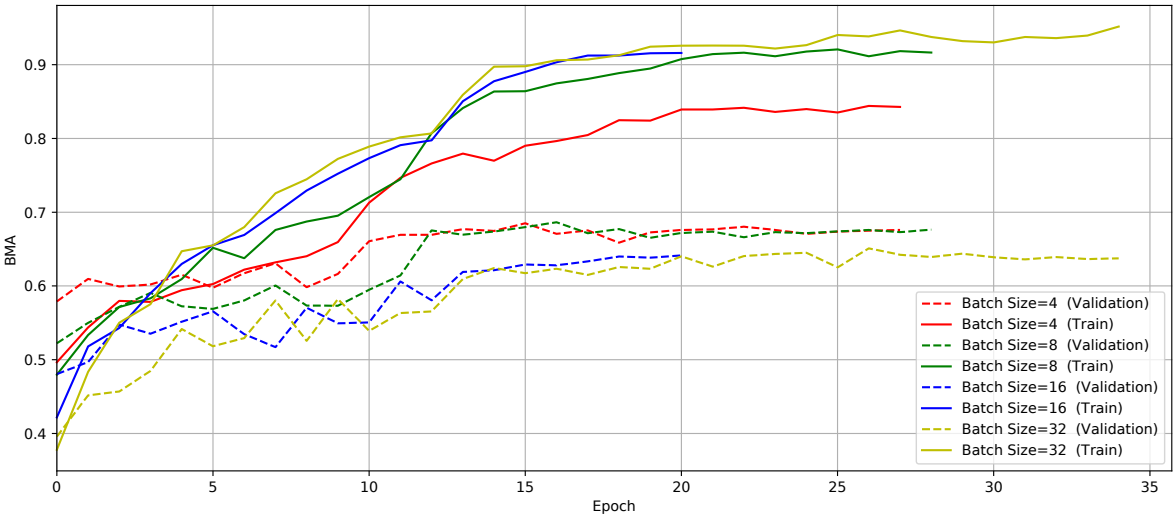

Figure 5.5: Batch size vs train and validation BMA over epochs for the fine-tuned DenseNet201 trained with 5000 samples.

5.2.2 Varying the Number of Epochs Before Fine-Tuning

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. More specifically, in one epoch all image samples are fed to the network in several iterations given by Equation 5.1. Therefore, the number of epochs defines how many times each image is seen by the network. The maximum number of epochs has been fixed to 100, but most of the tested models never reach that point because of early stopping.

In this approach to transfer learning, the pre-trained model's classifier is replaced by an untrained one, which means that the classifier's weights are initially defined by the Glorot's initialization method [34]. Knowing that the initial weights of the classifier are determinant for the overall model performance, one can hypothesize that by freezing the convolutional base for an initial number of epochs and only train the classifier weights for that amount of time, will lead to better converge point during the training process. Therefore, in this process, the classifier's weights are being adapted to the new dataset, before the actual fine-tuning process starts.

Figure 5.6 illustrates the impact of different numbers of epochs used to train the classifier before the fine-tuning process starts, on the overall performance. Results show that by using this method, models start to significantly improve BMA later, but generally train for more epochs and reach better converge points. For example, if one uses 0 epochs during this phase (immediately start the fine-tuning process), the model stops on epoch 20 in a rather worse convergence stop in comparison with using 2, 4 or even 6 epochs, which end the training process at the 31th, 26th and 26th epoch, respectively.

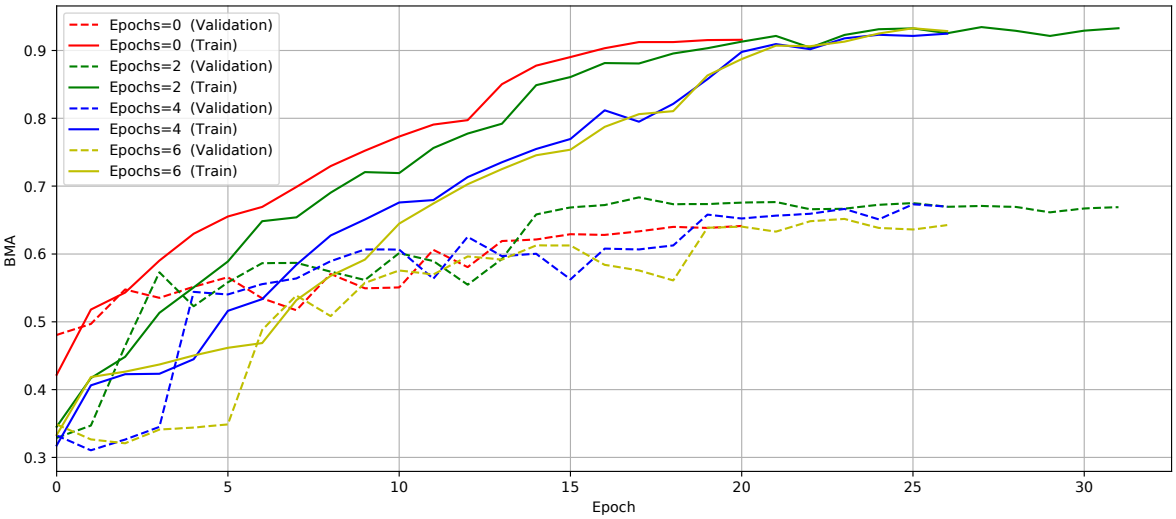

Figure 5.6: Number of epochs before the fine-tuning process vs train and validation BMA over epochs for the fine-tuned DenseNet201 trained with 5000 samples.

However, there are no significant improvements to be shown by using 4 or 6 epochs before the fine-tuning process starts in comparison with 2, as all of them end up with similar, if not worse, BMA scores. Therefore, the remaining experiments will use 2 epochs to train the classifier before the fine-tuning process of the convolutional base.

5.2.3 Varying the Learning Rate

Presumably, the most important hyperparameter to optimize is the learning rate. It controls the step size for model weight updates concerning the loss function. Lower values mean slower travel along the downward slope but consequently take more epochs to converge into a local minimum. In this approach, two learning rates are considered, namely, the learning rate used before the fine-tuning process starts and the initial fine-tuning learning rate.

Some experiments have been done to define which learning rate to use during the phase in which only the classifier is trained, more precisely, before the fine-tuning of the convolutional base process starts. Results in Figure 5.7 show that the model benefits from having larger learning rates during this phase, which presumably stems from the fact that the classifier's weights are not adapted to the ISIC 2019 dataset. Therefore, one can assume that using larger learning rates to adapt the classifier to the new dataset will ultimately lead to a better converge point by the end of the training process. As such, the learning rate used to only train the classifier before the fine-tuning process starts will be 10^{-3} for the remaining experiments.

Furthermore, there is a far more important learning rate which one can call the initial fine-tuning learning rate and it 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. One can verify this hypothesis by comparing various fine-tuning learning rates in Figure 5.8.

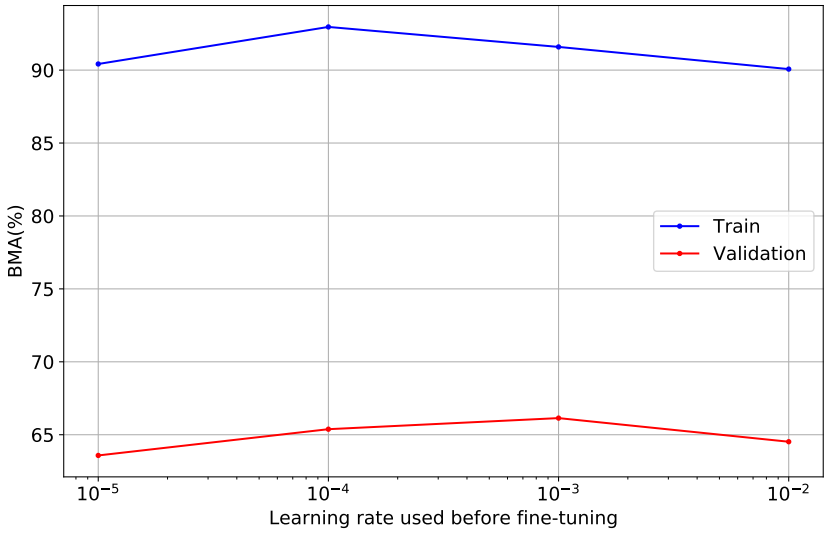

Figure 5.7: Learning rate used to train the classifier before the fine-tuning process vs BMA for the fine-tuned DenseNet201 trained with 5000 samples.

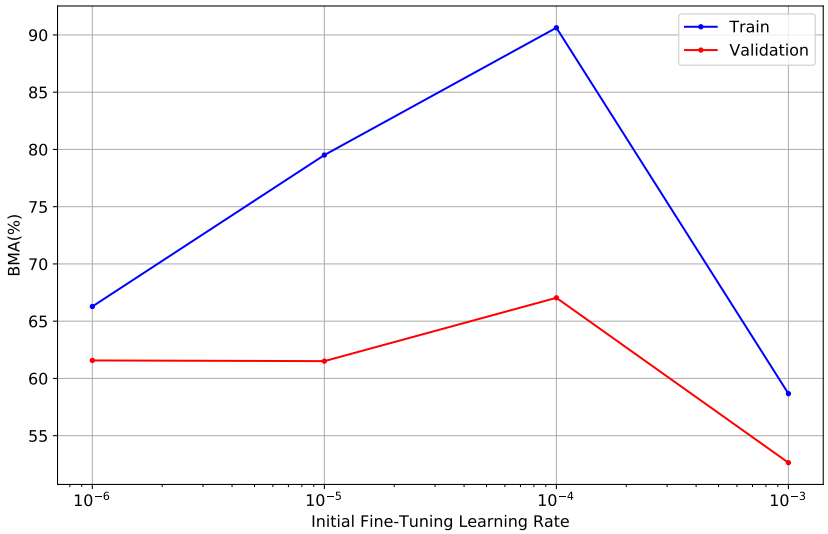

Figure 5.8: Fine-tuning learning rate vs train and validation BMA for the fine-tuned DenseNet201 trained with 5000 samples.

Figure 5.9 shows that a low learning rate like 10^{-6} makes the improvements linear, while higher learning rates like 10^{-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 10^{-3}. This is because there is too much "energy" in the optimization and the parameters are bouncing around chaotically, unable to settle in a good spot in the optimization landscape.

The learning rate of 10^{-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. After all, the rest of the epochs will be spent on minimizing the loss within that specific region, which is finding the local minimum. In contrast, learning rates like 10^{-5} and 10^{-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. Higher learning rates like 10^{-3} are too big, which leads to the network never finding a good region of search space at the beginning of the training process, and ultimately jumping between convergence stops.

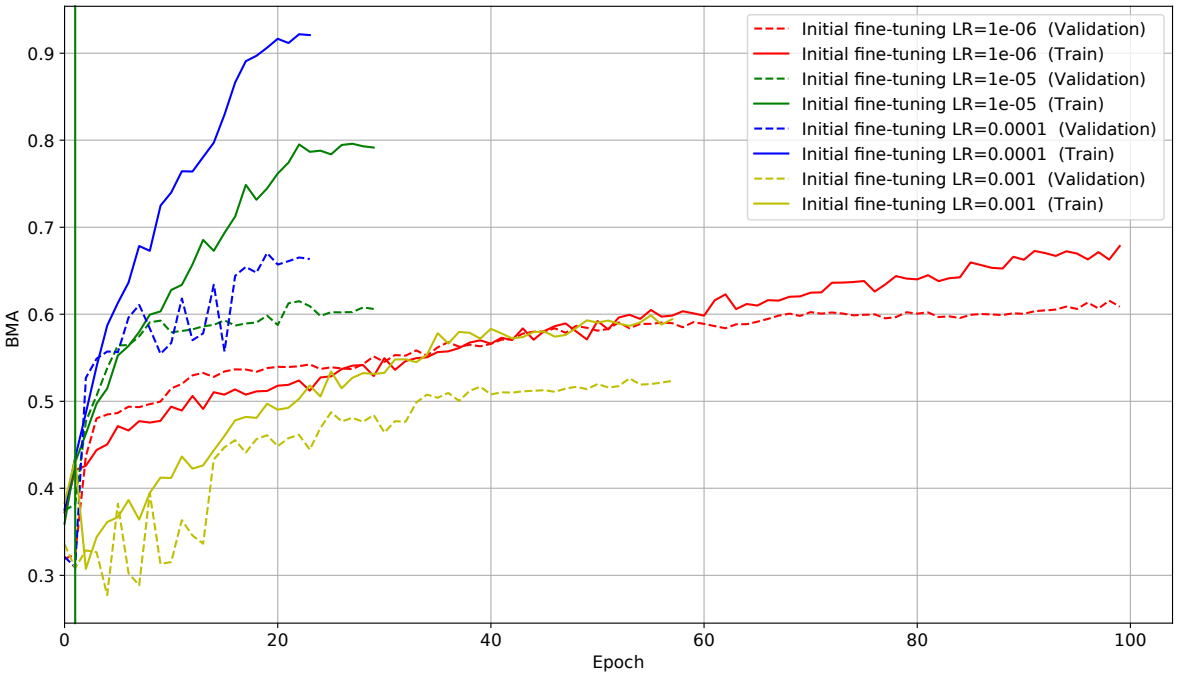

Figure 5.9: Influence of the fine-tuning learning rate on train and validation BMA over epochs for the DenseNet201 trained with 5000 samples.

One can also observe the influence of the learning rate scheduler by looking at Figure 5.10. A low learning rate like 10^{-6} is not affected by the learning rate scheduler as it keeps the same global learning rate across the whole training process. This happens because the validation loss keeps slowly decreasing (see Figure 5.11) until it reaches the maximum number of epochs. In contrast, high initial fine-tuning learning rates like 10^{-3} have a step curve evolution of the learning rate coming down until it reaches a local minimum at 10^{-3}. However, using such a learning rate will make the weight updates overshoot good convergence points as seen by the highs and lows of the validation loss curve.

Moreover, even though the learning rate of 10^{-4} does not decrease in a step-like manner like 10^{-3} (see Figure 5.10), it converges much more rapidly to a local loss minimum earlier during training as seen by the loss curve in Figure 5.11. The initial fine-tuning learning rate of 10^{-5} can also be a good option as it converges quickly to a similar loss point. However, considering the results Figure 5.8, it is clear that 10^{-4} is the better option, so it will be used for the remaining experiments.

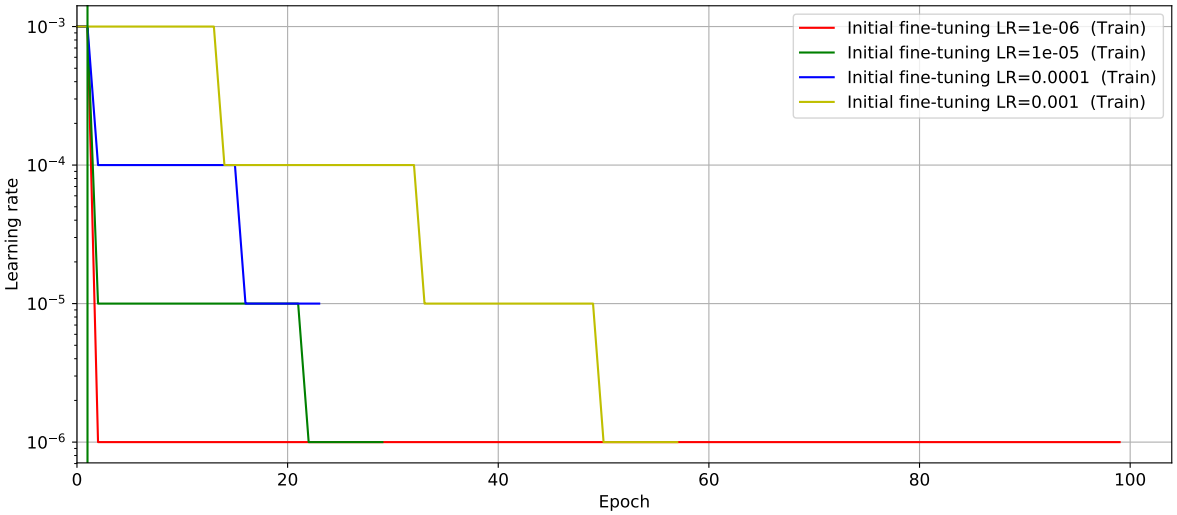

Figure 5.10: Initial fine-tuning learning rate vs the learning rate over epochs for the DenseNet201 trained with 5000 samples.

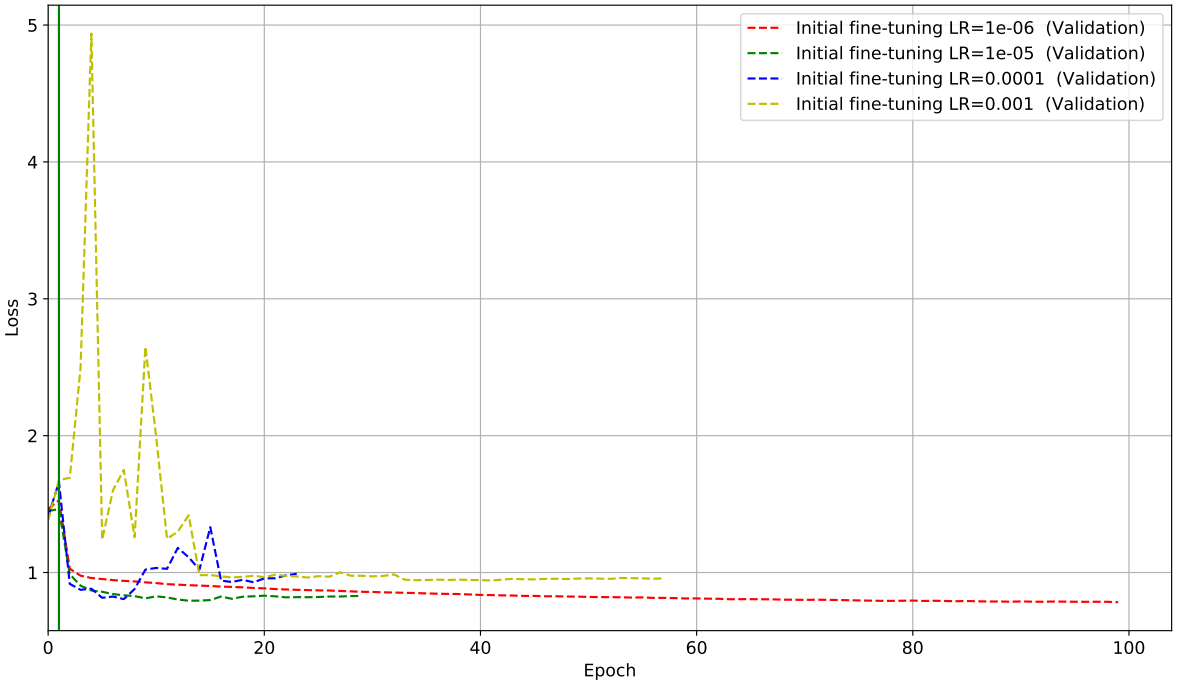

Figure 5.11: Initial fine-tuning learning rate vs validation loss over epochs for the DenseNet201 trained with 5000 samples.

5.2.4 Varying the Learning Rate Scheduler's Patience

During the training of deep neural networks, it is useful to reduce the learning rate as the number of training epochs increases. The intuition behind this concept is that with high learning rates, the deep learning model will make larger parameter updates. This is helpful during the early epochs because the model is searching for a convergence spot within a large number of possible combinations of parameters. However, in the latter stages of training, the intention is to find the optimal spot within a narrower search area rather than search

for other convergence spots. Therefore, the parameter search during the latter stages of the optimization process is supposed to have a much narrower search domain than in the earlier stages.

The learning rate schedulers usually have a hyperparameter called patience, which dictates how many epochs the training process should wait after no improvements were made within a predefined metric value. In this approach that metric is the validation loss, meaning that if the model does not decrease the validation loss for several epochs (patience), then the learning rate is decreased by a factor of 10. However, there is no clear value for the patience one model should have for the learning rate scheduler. It is hypothesized that high patiences will lead to chaotic improvements throughout the training process, but low patiences will eventually lead to premature early stopping.

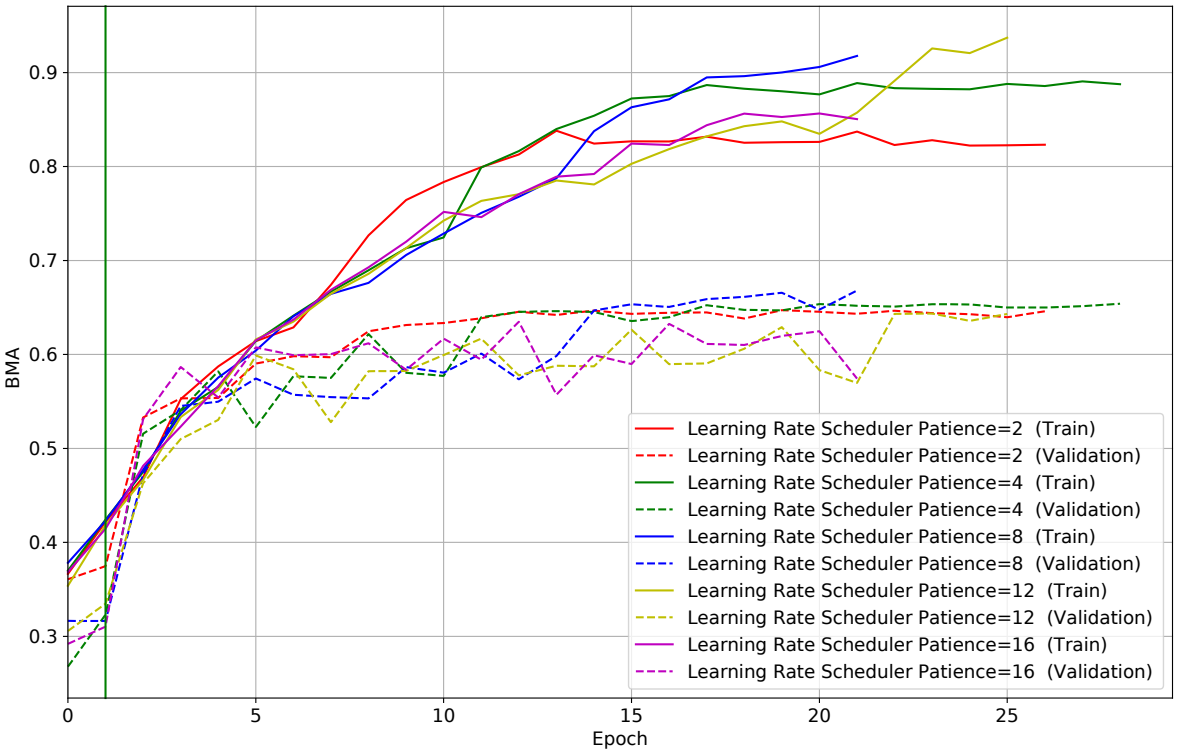

Figure 5.12: Influence of the learning rate scheduler's patience on train and validation BMA over epochs for the DenseNet201 trained with 5000 samples.

Experimentation with different patience values shows that if a model does not employ a learning rate scheduler scheme, it will end up in a worse convergence stop. By looking at Figure 5.12, one could observe that with the patience of 16, the train and validation BMA curves are more chaotic in comparison with lower patience values like 8. In this approach, a patience of 16 is equivalent to not using a learning rate scheduler (see Figure 5.13), because the early stopping happens at the 16th epoch.

Results in Figure 5.12 confirm the presented hypothesis. Patiences of 2 and 4 prematurely decrease the learning rate (see Figure 5.13), which makes the improvements less chaotic throughout the training process, but end up in with worse BMA, because earlier epochs were not able to find good convergence stops. In contrast, high patiences like 12 are not able to

find good convergence points because for most of the training process a learning rate of 10^{-3} is used. Finally, the patience of 8 epochs seems to be the sweet stop, with big improvements on earlier epochs but smaller and smoother improvements on latter epochs, representing a more narrow search of parameters, ultimately leading to a faster convergence process and a better BMA score.

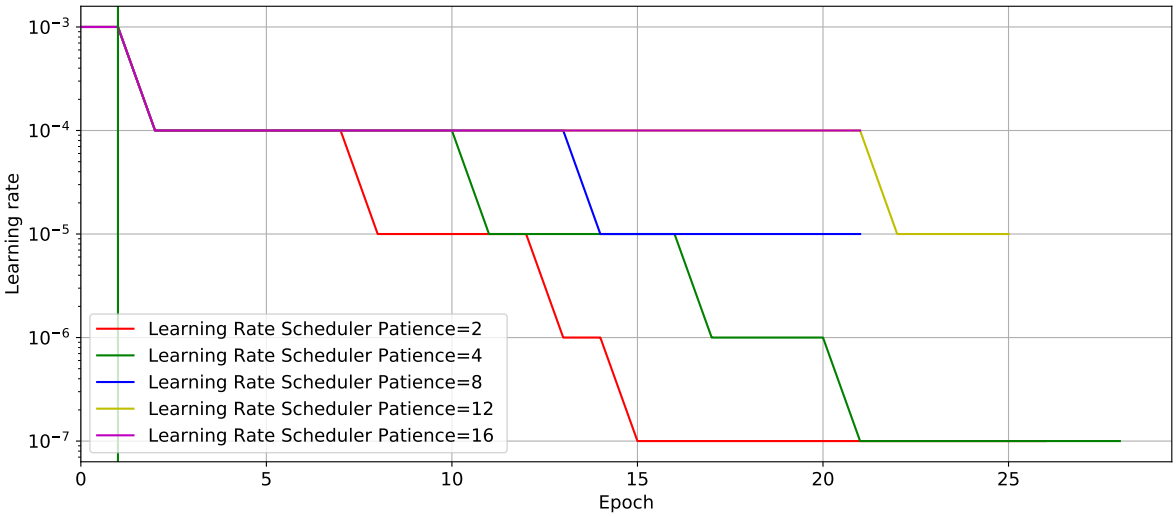

Figure 5.13: Influence of the learning rate scheduler's patience on the learning rate over epochs for the DenseNet201 trained with 5000 samples.

5.2.5 Applying Regularization Techniques

From the results in Figure 5.12, 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 BMA. In light of the problem, L2 regularization can help reduce this issue by adding a cost to the loss function of the network for large weights values [56]. Presumably, L2 regularization will create an overall simpler model that will be forced to learn only the relevant patterns in the training data.

However, results in Figure 5.14 show that the L2 regularization method has a negative impact on the train and validation BMA scores. Even though high L2 values reduce overfitting, they also significantly cut the performance values both on train and validation sets. Moreover, small L2 values do not significantly reduce overfitting, while still having a big impact on the generalization performance. Therefore, this method does not seem to be appropriate to reduce overfitting.

Another regularization technique called dropout [57] was also experimented with. For this approach, the classifier uses a dropout layer within the classifier, in which the dropout rate is varied. However, results in Figure 5.15 show that adding this method will significantly decrease the train and validation BMA. Furthermore, results show that no less overfitting is presented when applying such a technique. Therefore, similarly to the L2 regularization, this method does not seem appropriate to solve this overfitting problem.

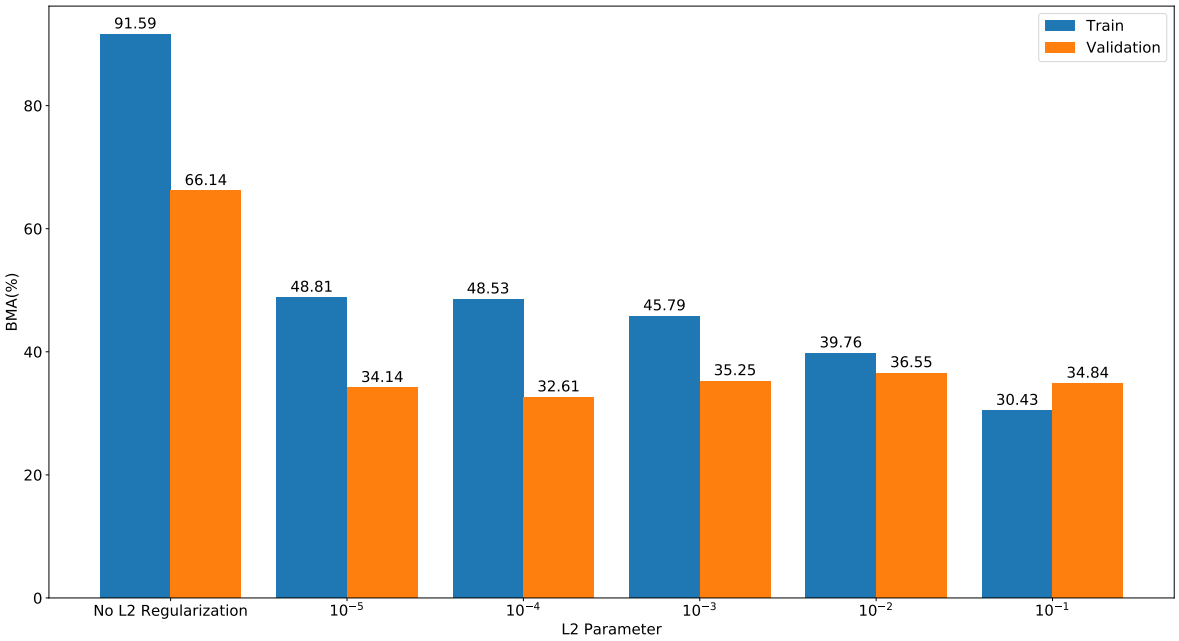

Figure 5.14: L2 regularization parameter vs train and validation BMA for the fine-tuned DenseNet201 trained with 5000 samples.

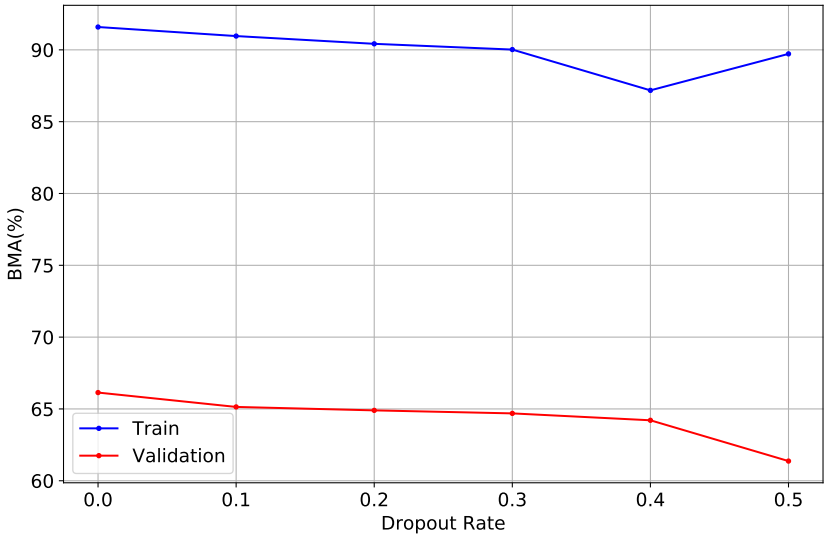

Figure 5.15: Dropout rate vs train and validation BMA for the fine-tuned DenseNet201 trained with 5000 samples.

5.3 Results Discussion

The results of the experimentation of different pre-trained models in Section 5.1 found clear support for the presented hypothesis. More specifically, applying transfer learning through fine-tuning the whole set of weights from both the convolutional base and the classifier is a better approach than total parameter extraction without fine-tuning in the context of skin lesion diagnosis. The argument behind these findings is that the dataset used to train those pre-trained models is largely different from the ISIC 2019 dataset, which means that those weights need to be adapted for the new dataset. Therefore, these results can be taken as a

heuristic for future adaptations of pre-trained models for skin lesion diagnosis.

Furthermore, results show that more recent CNN architectures such as DenseNet or EfficientNet (see Table 2.1) outperform older architectures like VGG or ResNet. Even within the same architecture, variants benefit from the extra number of layers and the extra number of trainable parameters for train and generalization performance. In comparison, pre-trained models like the VGG16 fall behind, presumably because these models lack the ability to create as many levels of abstraction as other networks with more layers.

Moreover, by extracting and fine-tuning the parameters of the convolutional base of the DenseNet201 model trained with a new classifier on top, the model attains a BMA score of 0.579 and a accuracy score of 0.746 (see Figure 5.3). However, as shown in the confusion matrix of Figure 5.16, even this model, which is the best performing pre-trained model, struggles on classification of underrepresented classes such as dermatofibroma (DF), while performing well with overepresented classes such as the melanocytic nevus (NV).

Figure 5.16: Confusion matrix of the fine-tuned DenseNet201 pre-trained model, trained on 5000 ISIC 2019 samples.

While hyperparameter tuning in Section 5.2 had a small impact on generalization performance, it allowed us to identify key insights about the influence of some of the network's hyperparameters. More specifically, results show that low batch sizes have a regularization effect on the training process, but in turn, extend the training times further. The presented results on the classifier weight updates show very little substantial impact when no updates are applied before the fine-tuning process, which hints that the pre-trained model's classifiers are well adapted to be initialized with conventional methods like Xavier's initialization.

Additionally, the importance of the start fine-tuning learning rate was clearly shown in Section 5.2.3. Too high fine-tuning learning rates overshoot convergence points and too low

learning rates make the weight updates too small which ultimately leads to slow improvements in the training process. However, the learning rate of 10^{-4}, makes the models converge fast concerning the loss function, which ultimately leads to better BMA scores on the validation set. Moreover, one should take into consideration the use of the learning rate scheduler as a method to adapt the learning rate along the training process. In this context, low patiences lead to premature early stopping, high patiences never find good convergence points along with the loss function, but a middle point (*e.g.*, 8) helps the model in the process of finding good convergence spot.

Finally, from the presented results, it is very apparent that there is a high amount of overfitting. Unfortunately this analysis found evidence that regularization methods such as the L2 regularization or dropout are quite ineffective for the undersampled dataset. Even with the hyperparameter tuning, the DenseNet201 model is only able to attain an accuracy of 75.4% and a BMA of 59.1% on the test set. The values of the BMA are low relative to the accuracy because underrepresented classes have poor performance in comparison with overrepresented classes (this effect is shown in the confusion matrix of Figure 5.17), which the accuracy metric does not take into account. Therefore, there is a clear need to improve the model's generalization performance, which is the issue that is going to be addressed in Chapter 6.

Figure 5.17: Confusion matrix of the hyperparameter optimized and fine-tuned DenseNet201 pre-trained model, trained on 5000 ISIC 2019 samples.

CHAPTER 6

Improving Model Generalization Performance

This chapter is divided into three different sections which will address the problems left open in Chapter 5, namely, the low generalization performance on the validation and test sets in comparison to the training set. For this purpose, Section 6.1 studies the impact of dataset size, data augmentation methods, and class balancing techniques. Subsequently, Section 6.2 attempts to improve this approach's performance by finding a suitable way for ensembling multiple models from different architectures. Afterward, Section 6.3 presents results of different procedures to deal with the out of the training distribution test images. Finally, Section 6.4 discusses and compares the obtained results to other state-of-the-art skin lesion diagnosis classifiers from the ISIC 2019 challenge.

6.1 Data Study

This section will explore the influence of factors like dataset size, data augmentation, and class balancing in the overall generalization capability of different variants of DenseNet pre-trained models. For each experiment, a benchmark has been set taken into consideration the results obtained in Chapter 5. More specifically:

- Pre-trained models from the DenseNet architecture will be used, as they are part of one of the best performing architectures in this dataset (see Figure 5.3), while also having a comparatively small number of trainable parameters for all of its 3 tested variants, namely, the DenseNet121, DenseNet169, DenseNet201 (see Table 2.1);
- An global average pooling layer is introduced at the end of the convolutional base in order to reduce the number of parameters for the classifier;
- The original pre-trained model's classifier is replaced by a new classifier composed of one fully-connected layer with 512 neurons which use the ReLU activation function, and one softmax layer with 8 neurons to translate each of the class's probabilities;

- For the transfer learning approach, all the layer's parameters from the convolutional base of the pre-trained model are extracted and fine-tuned, which yields the best performance for the ISIC 2019 dataset (see the results presented in Section 5.1);
- Following the work done by Gessert *et al.* [20] the Adam optimizer [33] is used, with $\rho_1 = 0.9$ and $\rho_2 = 0.999$;
- The convolutional base is frozen for the initial 2 epochs with a learning rate of 10^{-3}. The initial fine-tuning learning rate is 10^{-4}, but is reduced by a factor of 10 when validation loss stops improving for 8 epochs. Each model trains for a maximum of 100 epochs but early stopping is performed whenever validation loss stops improving for 16 epochs;
- For each epoch, all the samples are shuffled before being feed into the network;
- Each batch is composed of 8 samples;
- For each training process, 3 models are saved: The model that obtained the highest BMA on the validation set, the model with the lowest loss on the validation set, and finally the resulting model from the last epoch. However, the performance on the test set is evaluated according to the model which attained the highest BMA on the validation set.

6.1.1 Impact of Dataset Size

DenseNet201 has a considerably large number of trainable parameters when compared with other models such as DenseNet121 (see Table 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 more generalizable patterns within the training data. To put this into test, results for variations of the DenseNet with a smaller number of parameters, specifically, DenseNet121 and DenseNet169, were compared with DenseNet201 in Figure 6.1. Results show that indeed lowering the network's capacity reduces the amount of overfitting, but not by a significant margin.

Moreover, the DenseNet121 model in Figure 6.1 still shows a considerable amount of overfitting, as the validation BMA does not approximate the train BMA. 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, more specifically, 5000 training samples that keep the same class distribution as the original ISIC 2019 challenge dataset. This means that underrepresented classes like dermatofibroma have too few samples to learn strong generalizable features (see Figure 4.4).

Indeed, the results in Figure 6.2 show that an increase in dataset size enables the DenseNet201 model to attain an significantly better validation BMA score. So far, this has been the only comparison that was able to significantly improve generalization performance and reduce overfitting, corroborating the importance of the dataset used for training deep neural networks. However, it should be noted that even with the full dataset (20517 training samples), overfitting remains an issue that should be addressed.

Furthermore, by observing Figure 6.3, one can see that models trained with more samples take longer to converge. Presumably, this is related to the learning rate scheduler, because in

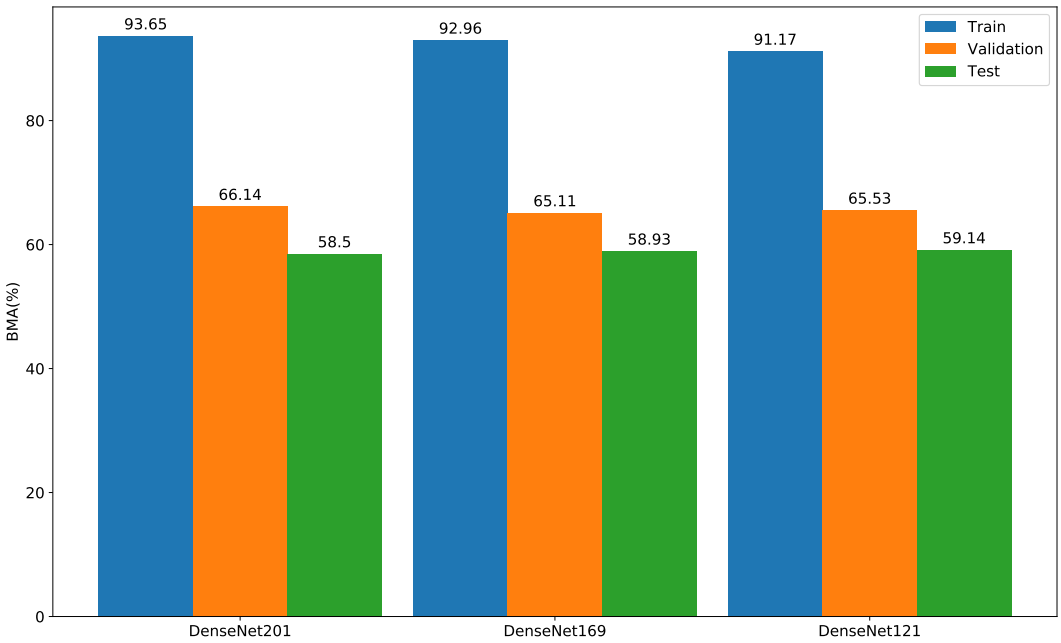

Figure 6.1: Different hyperparameter tuned DenseNet pre-trained models trained with 5000 samples vs BMA on train, validation and test sets. Online data augmentation is turned on.

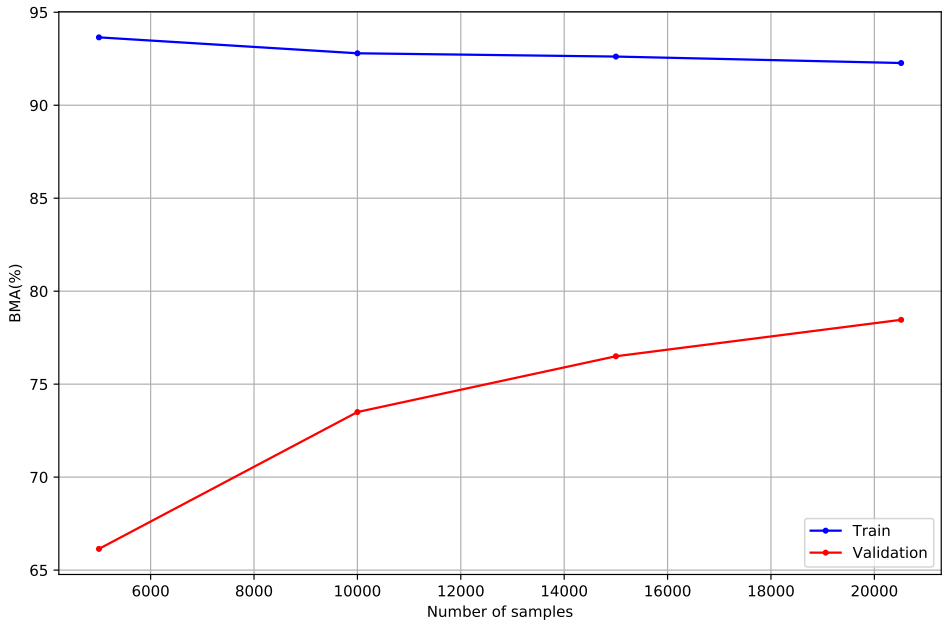

Figure 6.2: Number of training samples vs train and validation BMA for the DenseNet201 pre-trained model. Online data augmentation is turned on.

models with more samples, validation loss keeps improving for longer with high learning rates (*e.g.*, 10^{-4}), which makes the scheduler decrease the learning rate later (see Figure 6.4). In contrast, models trained with fewer samples stop improving the validation loss earlier, which makes the learning rate scheduler decrease the learning rate earlier. Consequently, as models tend to convergence to a local minimum with low learning rates, models trained with more samples take longer to converge because they take longer to reach low learning rates.

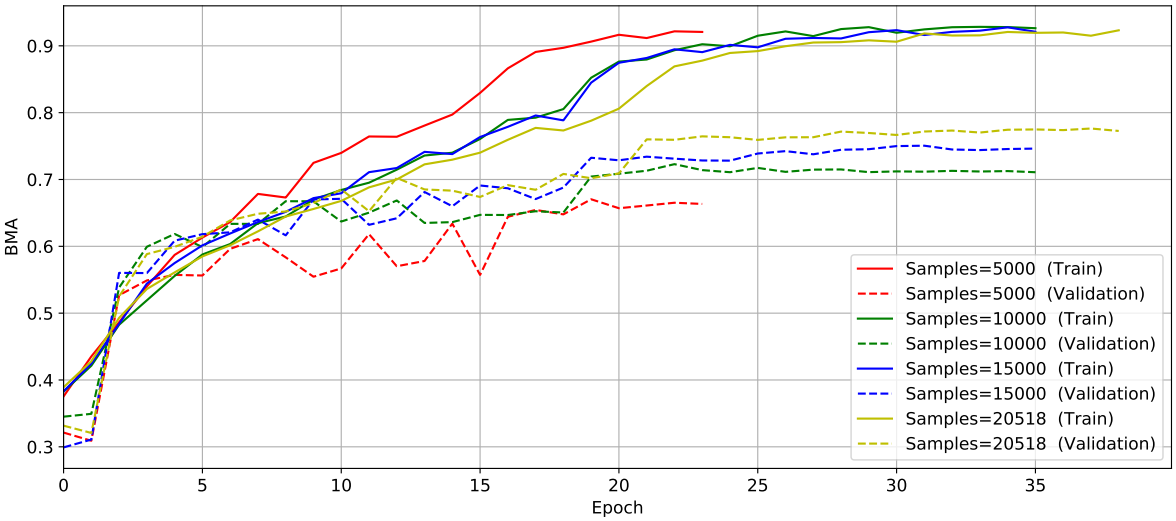

Figure 6.3: Number of training samples vs train and validation BMA over epochs for the DenseNet201 model. Online data augmentation is turned on.

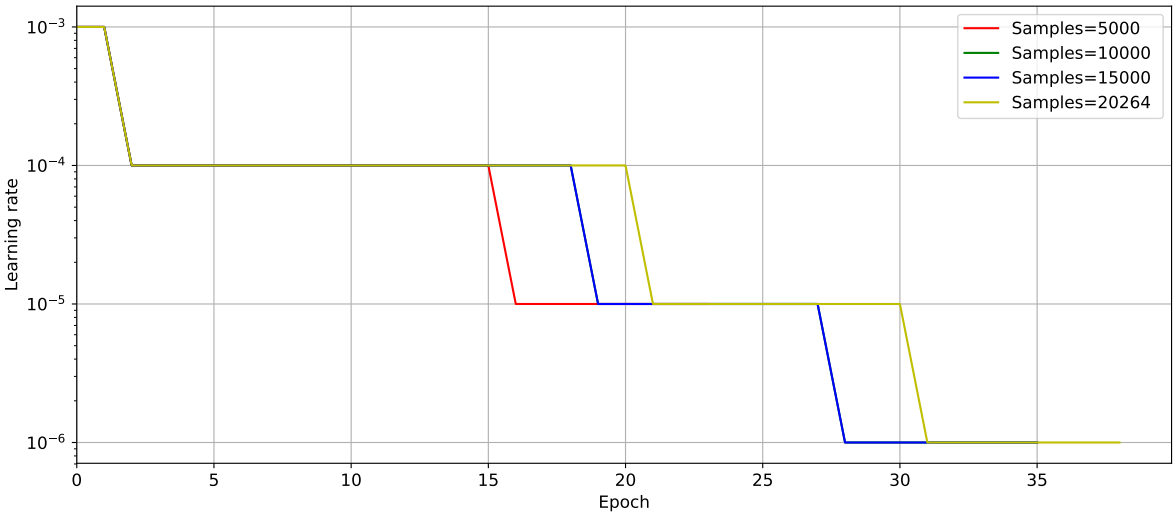

Figure 6.4: Influence of the number of samples on the learning rate over epochs for the DenseNet201 model.

The results from the plot graph in Figure 6.1 show that one could use a smaller pre-trained model from DenseNet, while maintaining similar performance, which would overall be beneficial to reduce overfitting. However, with a larger dataset, one could argue that the gap between models with a high and low number of trainable parameters would increase. Therefore, an experiment with different variations of the DenseNet has been made with 20517 training samples, to assess if this hypothesis is correct.

Nevertheless, results in Figure 6.5 show no significant improvements between different variants of the DenseNet architecture. Even though DenseNet201 got the best performance in comparison with its smaller versions, this improvement is not substantial as the difference between the DenseNet121 and DenseNet201 on test data is about 0.08%. Considering the results, that difference becomes even more irrelevant when taken into consideration that

the BMA gap between train and test on DenseNet121 is lower when compared with both DenseNet169 and DenseNet201. As such, the smaller model DenseNet121 proves to be a better choice as it slightly reduces overfitting while keeping similar performance to bigger models on the test set.

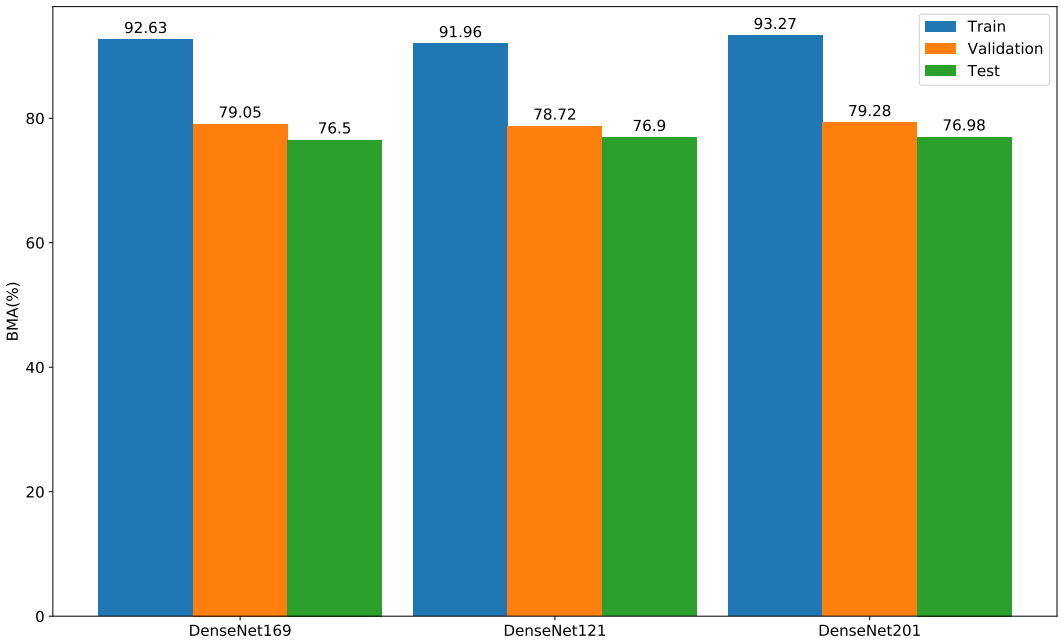

Figure 6.5: Different hyperparameter tuned DenseNet pre-trained models trained with 20518 samples vs BMA on train, validation and test sets. Online data augmentation is turned on.

Finally, with the full dataset, the DenseNet121 model is able to attain a BMA of 76.9% and a accuracy of 84.9 % over the test set. Furthermore, results show considerable improvements on the confusion matrix in Figure 6.6 when compared with the results from Figure 5.17. Overall every single class attains better performance on the test set, but this effect is much more noticeable in classes with fewer samples. Presumably, these results could be further improved with a larger training dataset, which should be a focus point for future benchmark challenges of ISIC.

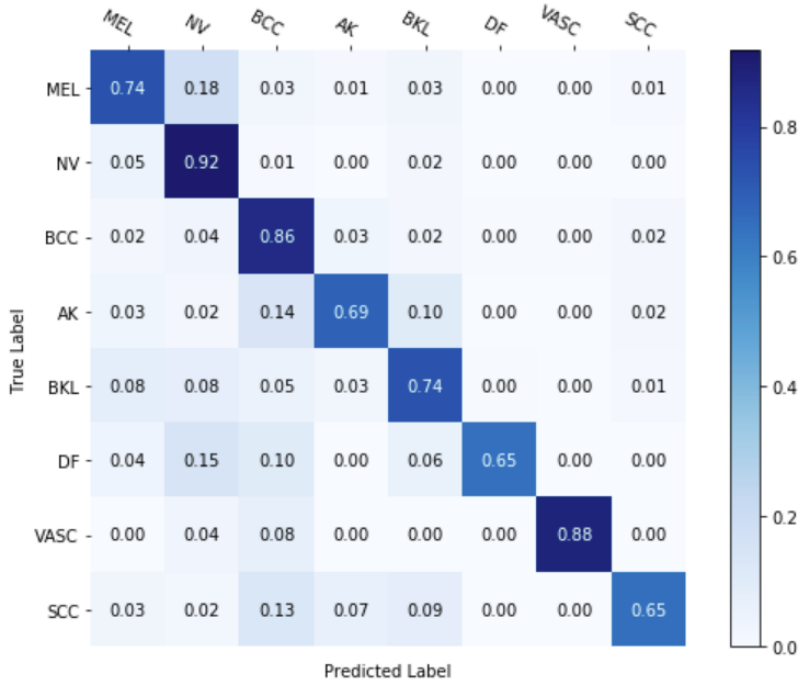

Figure 6.6: Confusion matrix of the hyperparameter optimized and fine-tuned DenseNet121 pre-trained model trained with the full unbalanced training dataset (20517 samples). Online data augmentation was turned on as a measure to reduce overfitting.

6.1.2 Impact of Augmentation Techniques

Considering the results from Figure 6.2, there is a considerable amount of overfitting even with the full training dataset. Moreover, results in Figure 6.6, still show a considerable lack of generalization performance for underrepresented classes. As such, different strategies of data augmentation can be used to help alleviate these problems. One of such strategies is called offline data augmentation, which attempts to create a larger dataset by generating synthetic samples through augmentation techniques before the training process starts.

However, there is a wide range of image processing methods for augmenting images (*e.g.*, rotations, distortions) and they can be combined to produce samples that are substantially 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, especially considering that some of them have different variations and parameters which can quite change the augmentation (*e.g.*, the magnitude of distortion, the angle of 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 6.7 shows a comparison between different augmentation groups applied to the DenseNet121 model trained with 20518 class balanced samples (approximately 2565 samples per class). Moreover, underrepresented classes are oversampled through the augmentation techniques from each of the groups, and overrepresented classes are undersampled by randomly choosing samples from that class without repetition. In turn, this means that 9089 samples are generated through data augmentation and the rest of the images are original (11420 samples).

Furthermore, the experiments use the augmentation techniques described in Section 4.4 which are spread across the different augmentation groups. More specifically:

- Group 0: No augmentation techniques are applied. For data augmentation as a class balancing measure (offline data augmentation), oversampling is done by randomly copying and pasting samples from a specific class with repetition. This is the baseline that other groups should be compared to;
- Group 1: Composed of slight variations to the original image, such as rotations of 90 degrees (clockwise), flips from top to bottom or vice versa, flips from left to right or vice versa, and crops that keep at least 85% of the original image pixels to assure that most of the lesion stays visible;
- 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 transformations. These transformations include tilts on either the left or right side, tilts forward and backward, skews from a random corner of an image, and finally, shears with a variation anywhere from 20 degrees towards the left to 20 degrees towards the right side of the image or anywhere from 20 degrees towards the top or 20 degrees towards the bottom of the 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 an 8x8 grid 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, ultimately producing noise in that area.

In groups 1, 2, 3, and 4 there is a 0.5 probability 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. Consequently, it could lead to problems like overfitting because the network would try to minimize loss for the same repeated samples, rather than learning generalizable knowledge from variations of that sample.

Considering the results from Figure 6.7, overall data augmentation does improve the performance in comparison with group 0, which does not employ any type of augmentation. Moreover, results indicate that simpler augmentation techniques like group 1 and 2 perform slightly better when compared to more complex augmentations groups such as group 3 and 4, which is to be expected in the context of skin lesion classification, because such groups alter important information about the lesion itself.

As described in Chapter 3, one of the methods used to identify skin lesions is the ABCDE method, which relies on features like the shape and color of the lesion to be diagnosed.

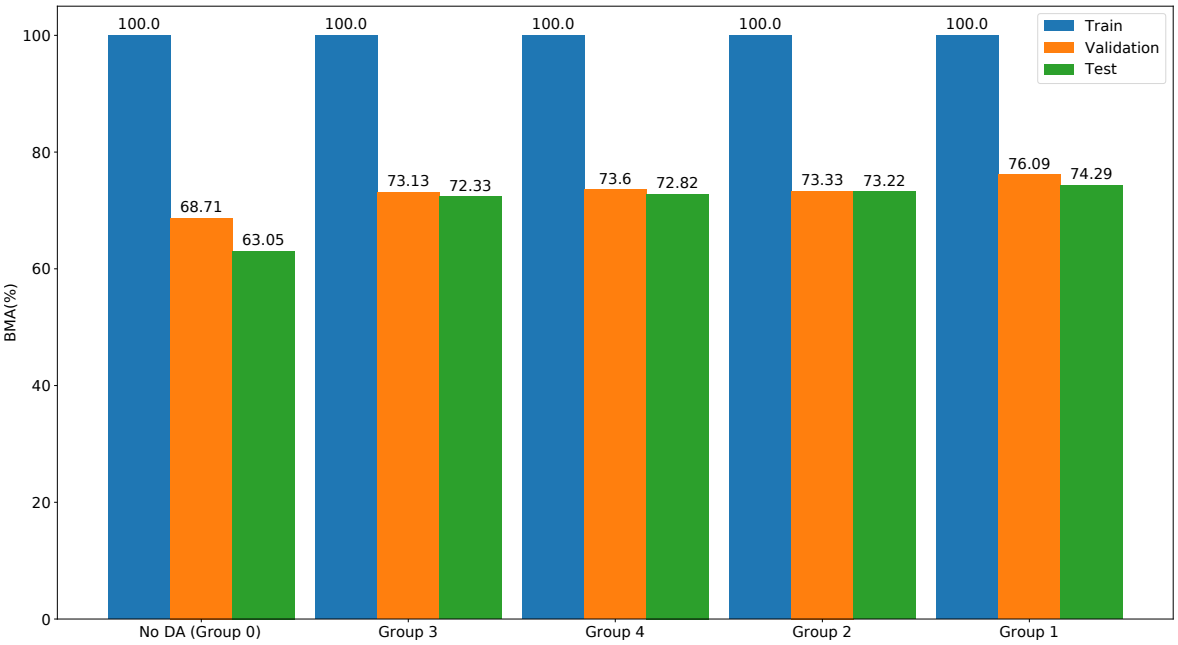

Figure 6.7: BMA of train, validation and test sets of different offline data augmentation groups with the DenseNet121 trained on 20518 balanced samples. Online data augmentation is turned off.

Therefore, by changing such features, one is potentially removing or changing important information about the lesion itself, which consequently will make the network perform worse. For example, group 4 introduces noise inducing transformations, however, one does not have the assurance that the random erasing will not erase important information from the lesion itself. Moreover, group 4 also applies distortions in the lesion samples, but such technique can also alter the shape of the lesion in such a way that it is no longer characteristic of its original lesion type. Similarly, by changing the colors of the lesion in group 2, important information about the color of the lesion might be changed, which is often an important property used by dermatologists to diagnose a lesion (*e.g.*, ABCDE method).

However, offline data augmentation is not the only method of applying data augmentation. Specifically, online data augmentation is from the literature presented in Chapter 3, a commonly used technique to reduce overfitting in deep learning based skin lesion classification models. While offline data augmentation can be used as a method to balance samples across classes, in online data augmentation, images are generated per iteration during training and differ in each epoch. In other words, in each epoch, the network is going to be fed a different variation of the original image, rather than the same image. Moreover, the results presented in Figure 6.7 show that turning off online data augmentation will cause a huge gap between train and validation/test BMA (overfitting), as the training BMA is 100% for all 5 groups.

Therefore, experiments were made to assess the impact of different combinations of online data augmentation and offline data augmentation (as a class balancing measure). As shown in Figure 6.8, online data augmentation significantly reduces the gap between train and validation and overall improves BMA on the test set, both when employed with and without

offline data augmentation. Even though the model with offline and online data augmentation performed better, it also increased overfitting in comparison with the model only trained with online data augmentation. Presumably, these results reflect the concerns discussed in Section 3.4, namely, data augmentation can lead to a model that easily discriminates synthetic examples but does not generalize the attained knowledge to real data [17].

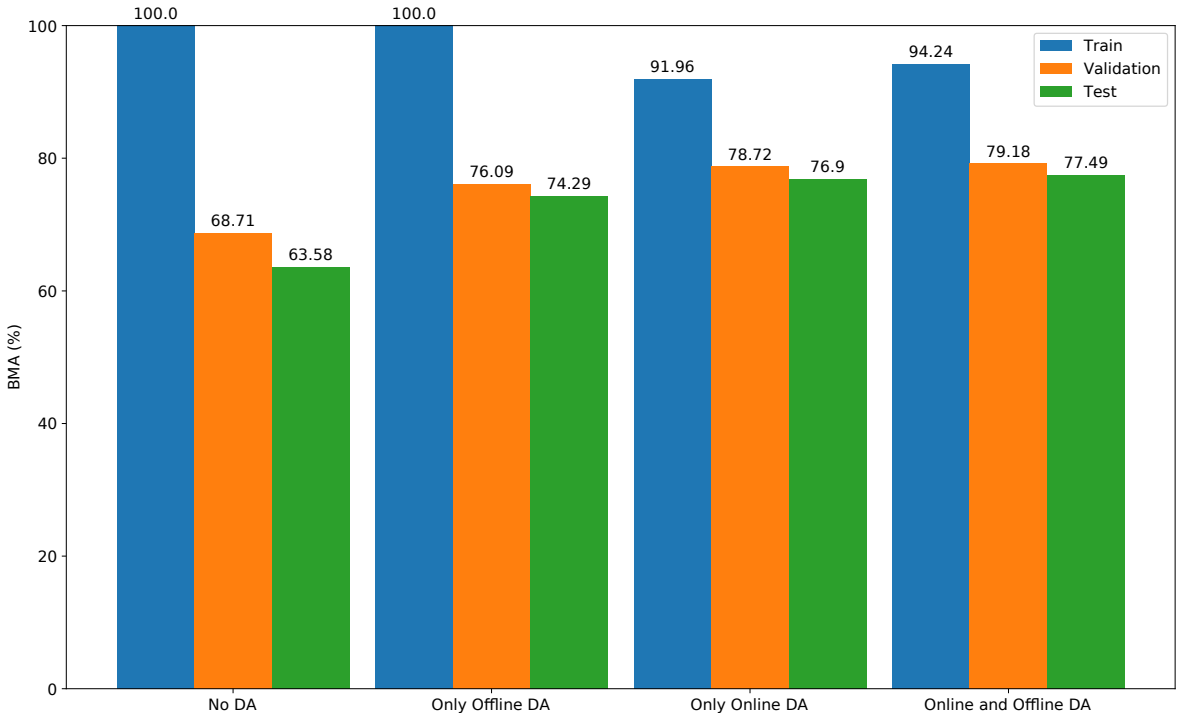

Figure 6.8: BMA of the train, validation and test sets of different combinations of offline and online data augmentation modes with the DenseNet121 trained on 20518 balanced samples. "No DA" is equivalent to data augmentation group 0 and "Only Offline DA" is equivalent to group 1 in figure Figure 6.7. "Only Online DA" simply employs online data augmentation during the training process. "Online and Offline DA" combines the class balancing measures from offline data augmentation and the overfitting measures from online data augmentation during training.

In the experiments showed so far, online data augmentation always uses augmentation group 1, which as shown in Figure 6.8, significantly reduces overfitting. However, an interesting question can be made of whether other augmentation groups can have the same effect on overfitting and generalization performance for online data augmentation. As such, an experiment with different augmentation groups for online data augmentation with an unbalanced dataset has been done, where the dataset used is the original 20518 samples to not bias the obtained results (no offline data augmentation).

Figure 6.9 shows that online data augmentation significantly reduces overfitting for all 4 augmentation groups. Furthermore, group 3 seems to almost eliminate overfitting, because the train, validation, and test BMA scores are quite similar. Group 3 also displays the best generalization performance on the test set in comparison with other groups. In contrast, group 1 performs slightly worse than group 3 on the test set, but significantly increases the amount of overfitting because the gap between train and validation/test is much larger. Other groups

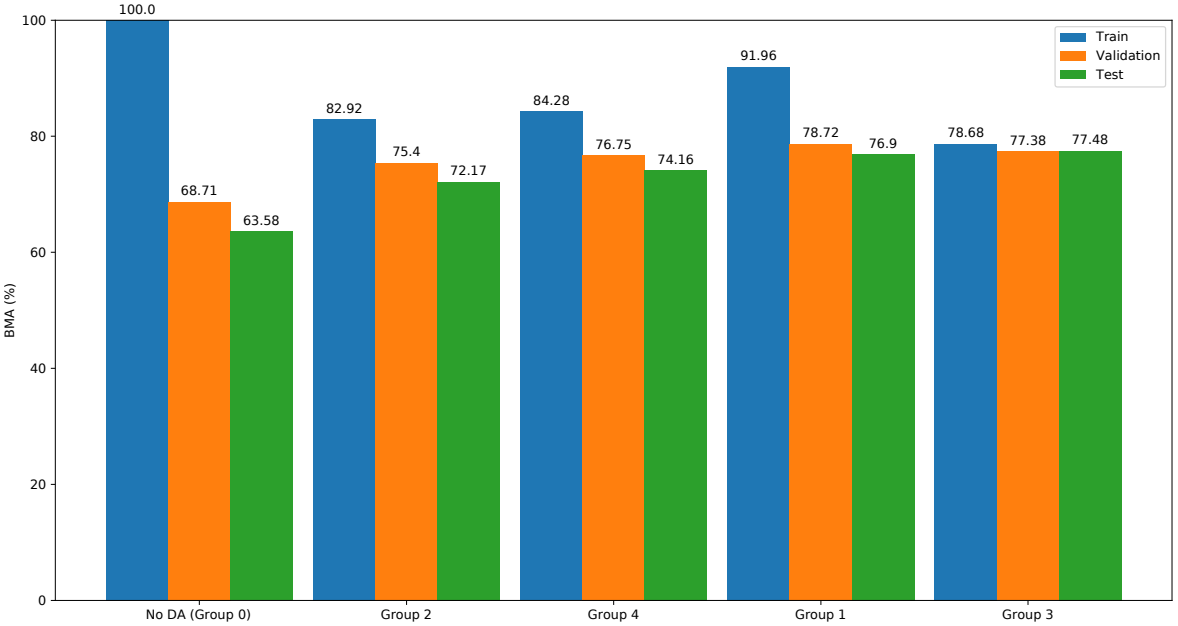

Figure 6.9: BMA of train, validation and test sets of different online data augmentation groups with the DenseNet121 trained on 20518 unbalanced samples (no offline data augmentation).

like 2 and 4 also reduce overfitting, but significantly reduce generalization performance on the test set.

Presumably, these results stem from the nature of online data augmentation. Considering an input image, in each iteration of the training process, online data augmentation will produce a different image. This means that samples generated by the augmentation procedures of group 3, will produce significantly different images in each iteration of the training process (see examples in Figure 4.11). In contrast, group 1 will produce somewhat similar images across all iterations because it is composed of simple augmentation techniques such as flips, rotations, and crops. This means that when one uses group 3 for online data augmentation during the training process, the model is obliged to learn a different feature each time a sample is fed to the network because it sees a significantly different sample in each iteration. Therefore, the model is forced to learn generalizable knowledge across these iterations rather than memorize equal or similar samples, like in augmentation group 1.

Unfortunately, these results came late in the development process of this work which meant that the next experiments were made using the augmentation group 1 for online data augmentation rather than group 3. Even though the BMA scores of group 1 and 3 are similar for the test set in Figure 6.9, the findings of using group 3 present a major way of reducing overfitting that should be further explored in future ISIC challenges.

6.1.3 Impact of Class Balancing through Data Augmentation

The results in Figure 6.6 show that even by using the full dataset, classification towards underrepresented classes (*e.g.*, squamous cell carcinoma or dermatofibroma) has overall worse performance than classes with more samples (*e.g.*, melanocytic nevus). Moreover, results in Figure 6.7 have shown that offline data augmentation as a class balancing measure significantly

improves the BMA scores across the train, validation and test sets. Therefore, one could hypothesize that this method can improve the generalization performance of the network further in underrepresented classes.

Results in Figure 6.10 show an experiment where a balanced model with 2533 samples per class is compared against an unbalanced model of 20518 samples by their BMA over the train, validation, and test sets. Results show that the balanced model will attain a better BMA score concerning all these three sets.

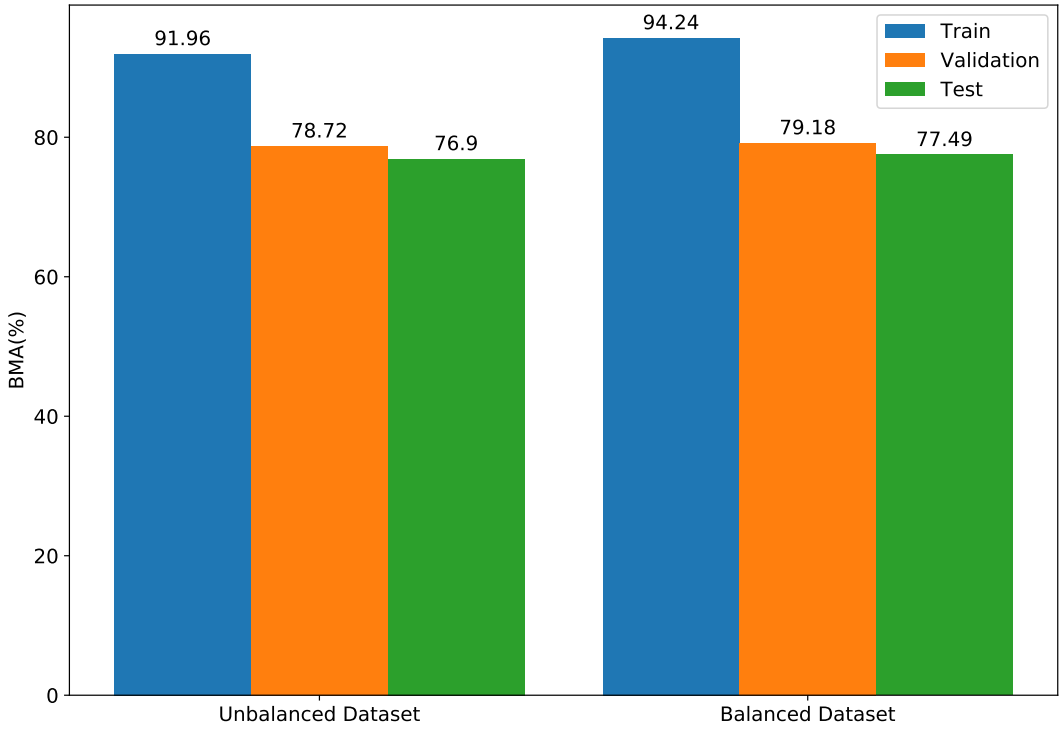

Figure 6.10: Comparison of the BMA of balanced and unbalanced datasets with 20518 samples for the train, validation and test sets using the hyperparameter optimized and fine-tuned DenseNet121 pre-trained model. Online data augmentation is turned on using group 1.

However, by doing the same comparison but with the accuracy metric, results show that the accuracy over the unbalanced dataset is better than those from the balanced dataset (see Figure 6.11). 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.

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 BMA scores. Figure 6.12 shows the influence of different balanced dataset sizes on the BMA and accuracy scores. Larger balanced datasets are created by decreasing the undersampling of original samples on overrepresented classes and increasing oversampling through offline data augmentation on underrepresented classes. Furthermore, experiments were made for 10000, 20000, 30000, 40000, 50000, 60000, 70000, and 83432 class

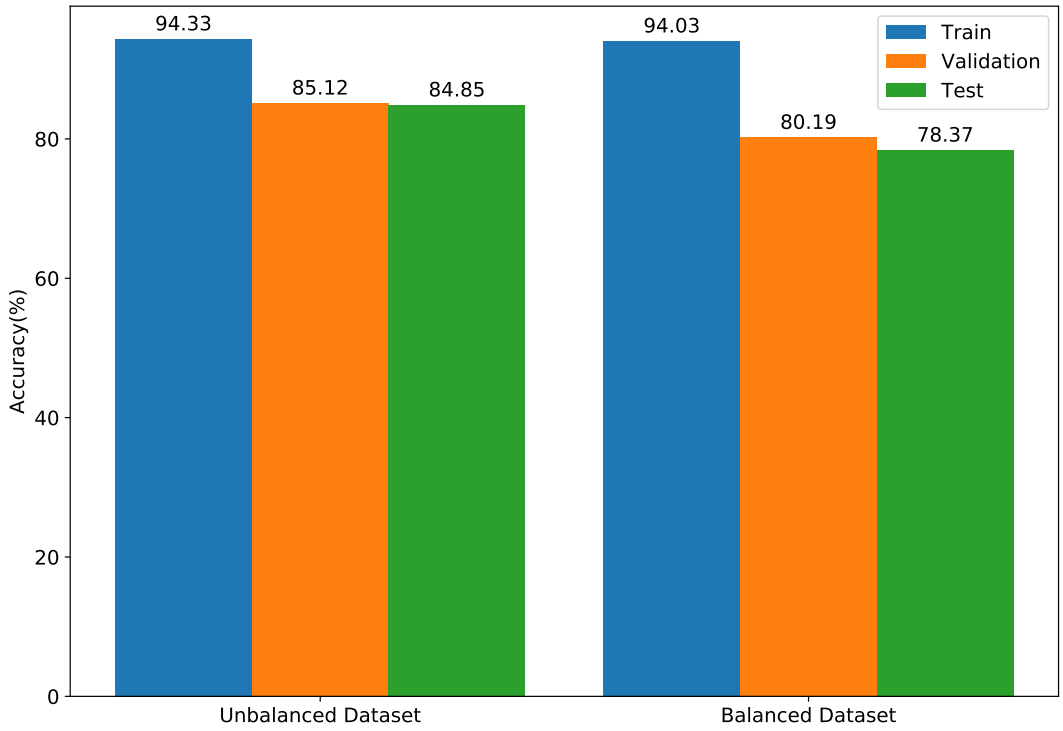

Figure 6.11: Comparison of the accuracy of balanced and unbalanced datasets with 20518 samples for the train, validation and test sets using the hyperparameter optimized and fine-tuned DenseNet121 pre-trained model. Online data augmentation is turned on using group 1.

balanced datasets, in which the dataset with 83432 samples has no undersampling being applied, meaning that no original samples are discarded. Therefore, each class of the dataset with 83432 samples has 10429 samples because the most overrepresented class in the training set is the melanocytic nevus (NV) with 10429 samples.

Considering the results from Figure 6.12, it seems that increasing the dataset through offline data augmentation does improve BMA and accuracy scores on both the test and validation sets. Furthermore, on the training dataset, the BMA scores approximate the accuracy scores, which is to be expected because the training dataset is balanced (recall Section 2.6).

The BMA and accuracy scores are close together on test and validation sets on smaller datasets such as with 10000, 20000, and 30000 samples. However, with larger datasets, the gap between BMA and accuracy scores start to increase. This gap is due to an overall increase in the generalization performance of overrepresented classes, while underrepresented classes do not keep improving. This happens because on smaller datasets overrepresented classes are undersampled, but most if not all of the underrepresented classes are already being oversampled. Therefore, performance towards overrepresented classes keeps rising, which greatly influences the accuracy scores, but underrepresented classes keep the same performance, which in turn makes a small impact on BMA scores. These results support the notion that past a certain number of synthetically generated samples, oversampling has close to no effect on generalization performance of underrepresented classes.

Furthermore, although oversampling the original dataset with class balancing seems to reduce overfitting, one can argue that this is due to the decrease in the undersampling of overrepresented classes, rather than the oversampling process. One can confirm such hypothesis by looking at the discrepancy of the accuracy and BMA on larger datassets.

It is also clear from the results that synthetic samples do not have the same impact on generalization performance as original samples. For example, the obtained improvements on the BMA scores shown in Figure 6.12 are considerably lower than the ones obtained in Figure 6.5 in which the dataset is increased with original samples rather than synthetic ones.

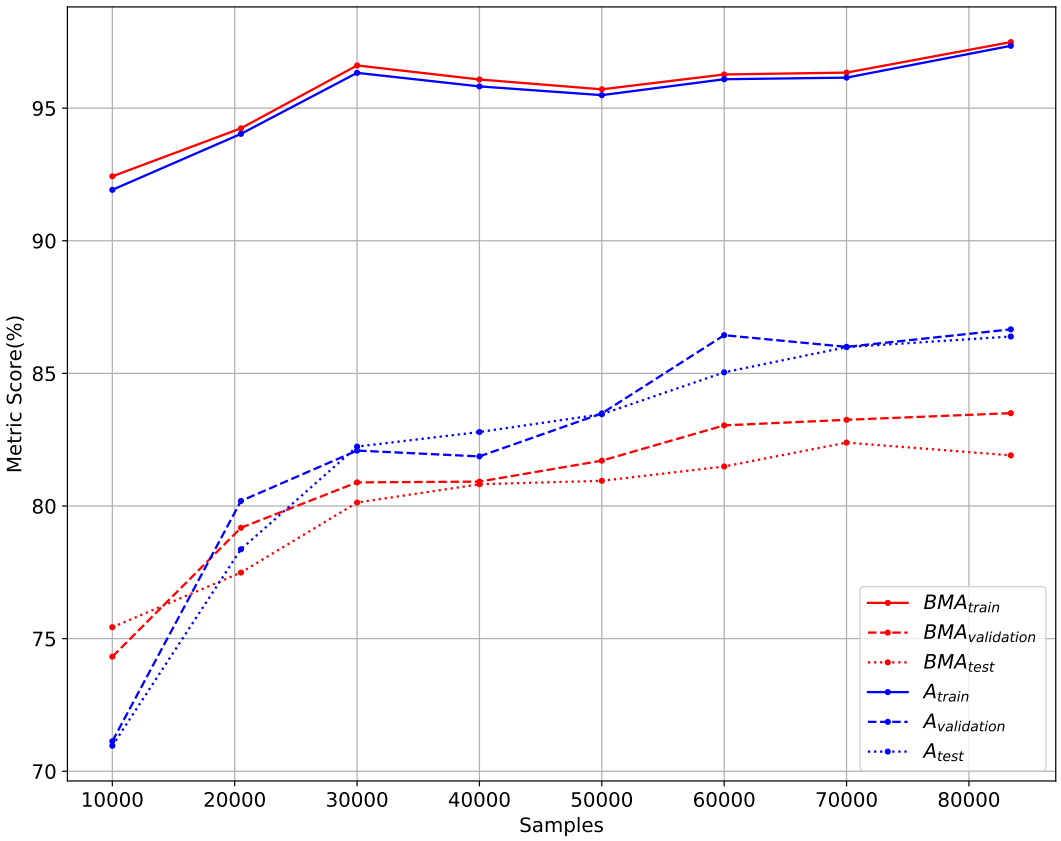

Figure 6.12: Comparison of train, validation and test BMA and accuracy scores for different balanced dataset sizes using the described oversampling and undersampling techniques. The pre-trained model used is the hyperparameter optimized and fine-tuned DenseNet121. Online data augmentation is turned on using augmentation group 1.

Finally, the DenseNet121 model trained with online data augmentation (group 1) on the class balanced dataset with 83432 samples is able to attain an accuracy of 86.70% and a BMA of 81.5% (see Figure 6.12). These results show the remarkable impact of online and offline data augmentation techniques on the generalization performance of the network. This represents a significant improvement from the results obtained in Section 6.1.1, where the model was not using class balanced samples. Presumably, this jump in performance is due to an increase in sample size for classes that are underrepresented on the original dataset. One can confirm such hypothesis by looking at Figure 6.13, which in comparison with Figure 6.6 shows that the network is more capable of correctly classifying underrepresented classes.

Figure 6.13: Confusion matrix of the hyperparameter optimized and fine-tuned DenseNet121 pre-trained model, trained with the balanced and oversampled ISIC 2019 dataset with 83432 samples. Online data augmentation was turned on during training with augmentation group 1.

6.2 Model Ensemble

The literature suggests that the most successful approaches towards ISIC challenges are those based on an ensemble of classifiers [14]. Even though this methods might not be practical for real world scenarios due to the added computational requirements, it might be useful to assess what improvements it could bring to the generalization performance of this approach.

As seen in Section 3.3.1, in order to properly create an ensemble one must combine different classifiers trained on different conditions (*e.g.*, model architectures, hyperparameters) or with different data (*e.g.* bagging). However, an ensemble of different models pre-trained on ImageNet will be created because in this study different pre-trained models were already studied in Section 5.1. Furthermore, an important aspect to consider is that simply selecting different pre-trained models within the same architecture is not enough because the different variations within a model architecture are just scaled up/down versions of a baseline model. Therefore, averaging the predictions of such models would not significantly increase the overall performance because each model would produce similar results (*e.g.*, VGG16, and VGG19 or EfficienNetB2 and EfficientNetB0).

Therefore, the choice of the models to use from each architecture in the ensemble is made based on the results from Figure 5.3. More specifically, this approach selects the best performing model from each architecture: VGG16, ResNet152, InceptionResNetV2 and EfficientNetB2. The exception to this criteria is the DenseNet, as the chosen model is the DenseNet121 instead of the DenseNet201 (see arguments behind this decision in Section 6.1.1).

Furthermore, different combinations between each of these models will be presented in order to attain the best possible performance. For simplification, all models will use the hyperparameters optimized for DenseNet in Section 5.2. All the models were trained under the same conditions based on the previous section's results, more specifically:

- An global average pooling layer is introduced at the end of the convolutional base in order to reduce the number of parameters for the classifier;
- The original pre-trained model's classifier is replaced by a new classifier composed of one fully-connected layer with 512 neurons which use the ReLU activation function, and one softmax layer with 8 neurons to translate each of the class's probabilities;
- Extracting all layer's weights while also fine-tuning these layers, which as shown in Section 5.1 yields higher performance since it will continue to optimize parameters relative to the target dataset;
- Following the work done by Gessert *et al.* [20] the Adam optimizer [33] is used, with $\rho_1 = 0.9$ and $\rho_2 = 0.999$;
- The convolutional base is frozen for the initial 2 epochs with a learning rate of 10^{-3}. The initial fine-tuning learning rate is 10^{-4} but is reduced by a factor of 10 when validation loss stops improving for 8 epochs. Each model trains for a maximum of 100 epochs but early stopping is performed whenever validation loss stops improving for 16 epochs;
- For each epoch, all the samples are shuffled before being feed into the network;
- Each batch is composed of 8 samples;
- Both online and offline data augmentation are used with random crops, rotations, and flips each with a probability of 0.5. Offline data augmentation is used as a class balancing measure and online data augmentation as a method to alleviate overfitting;
- For each training process, 3 models are saved: The model that obtained the highest BMA on the validation set, the model with the lowest loss on the validation set, and finally the resulting model from the last epoch. However, the performance on the test set is evaluated according to the model which attained the highest BMA on the validation set.

Results from Table 6.1 show significant improvements of all ensembles over single-model performance on both BMA and accuracy, with the ensemble of the best 3 models being the best performing one reaching near 90% accuracy. Presumably, all the ensembles have superior performance to single models, because one is averaging predictions across models with significantly different architectures. This produces significant variations within the softmax probabilities, which when averaged produce better generalization performance.

One can also observe that ensembling the most amount models is not necessarily the best approach. For example, the ensemble of the best 4 and the ensemble of 5 yields significantly worse performance than the ensemble of the best 3. This can be attributed to the discrepancy in the performance of VGG16 and ResNet152 concerning the rest of the models (EfficientNetB2, DenseNet121, InceptionResNetV2), which as a consequence of the averaging scheme brings the performance metrics down. However, the models composing the ensemble of the best 3 all have similar performance, which makes them good candidates for an ensemble with an

Approach Name	Pre-trained models	BMA	Accuracy
EfficientNetB2 approach	EfficientNetB2	≈ 0.820	≈ 0.851
DenseNet121 approach	DenseNet121	≈ 0.815	≈ 0.867
InceptionResNetV2 approach	InceptionResNetV2	≈ 0.811	≈ 0.862
ResNet152 approach	ResNet152	≈ 0.797	≈ 0.858
VGG16 approach	VGG16	≈ 0.753	≈ 0.824
Ensemble of best 2 models	EfficientNetB2, DenseNet121	≈ 0.842	≈ 0.878
Ensemble of best 3 models	EfficientNetB2, DenseNet121, InceptionResNetV2	≈ 0.846	≈ 0.891
Ensemble of best 4 models	EfficientNetB2, DenseNet121, InceptionResNetV2, ResNet152	≈ 0.841	≈ 0.894
Ensemble of all 5 models	EfficientNetB2, DenseNet121, InceptionResNetV2, ResNet152, VGG16	≈ 0.836	≈ 0.893

Table 6.1: BMA and Accuracy of different models and ensembles for 8-class classification of skin lesions. The Softmax probabilities a ensemble's models are averaged together in order to create each ensemble. All models (including models within ensembles) are fine-tuned and use the best hyperparameters found in Section 5.2. The training dataset is composed of 83432 class balanced samples (offline data augmentation) where synthetic samples are created using the augmentation group 1. Online data augmentation is turned on with augmentation group 1 as a measure to reduce overfitting.

averaging scheme. Moreover, one can assume that if more models were available with similar performance to the EfficientNetB2, DenseNet121, and InceptionResNetV2 then it would result in better BMA and accuracy scores.

Finally, the confusion matrix of the ensemble of the best 3 models is shown in Figure 6.14. Results show that overall every class has a slightly better performance when compared to the single-model performance of DenseNet121 in Figure 6.13. This performance increase also reflects on the BMA and accuracy values, which become 0.846 and 0.891, respectively.

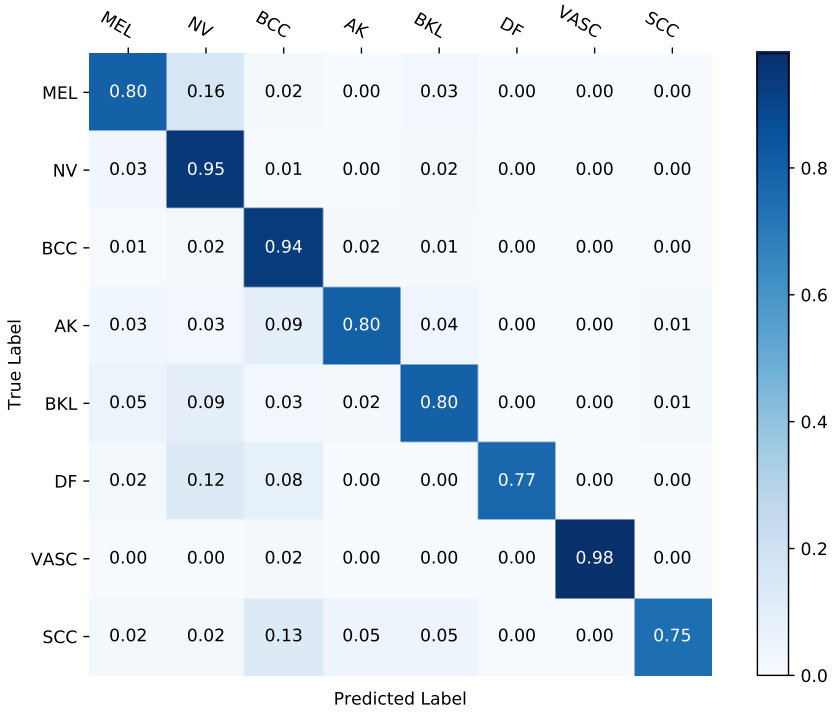

Figure 6.14: Confusion matrix of the ensemble composed with the best 3 pre-trained models, namely DenseNet121, EfficientNetB2 and InceptionResNetV2. Each model is trained with 83432 balanced samples with the best hyperparameters obtained in Section 5.2. Online data augmentation is turned on.

6.3 Out of Training Distribution Detection

The presented models in Section 6.1 and Section 6.2 show that deep CNNs can be effective predictors on the task of skin lesion classification. However, no out-of-distribution detection is being performed to identify "unknown" samples. In other words, the model is unaware of whether a test sample is part of the original train data distribution (*i.e.*, the original 8 classes) or not. As presented in Chapter 3, in the ISIC 2019 challenge, there is not a single way of identifying out-of-distribution samples, consequently resulting in a multitude of methodologies being employed by different authors. As a means to compare such methods, this work will explore 3 of them, namely:

- Following the submission by Zhou *et al.* [25], experiments will be made with different top-1 softmax probability thresholds;
- The ODIN [67] will be integrated within the single-model approach, following Wang's submission towards the ISIC 2019 [102];
- Finally, like Gessert *et al.* [21] (ISIC 2019 winner), results of out-of-distribution detection will be presented by training a model with an additional outlier class, composed of additional samples.

A benchmark has been set to test and compare these 3 approaches, more specifically:

- For simplicity of the results, the single best model approach towards 8 class classification, namely the hyperparameter tuned DenseNet121 model, will be used instead of the

ensemble presented at section 6.2;

- An global average pooling layer is introduced at the end of the convolutional base in order to reduce the number of parameters for the classifier;
- The original pre-trained model's classifier is replaced by a new classifier composed of one fully-connected layer with 512 neurons which use the ReLU activation function, and one softmax layer with either 8 or 9 neurons to translate each of the class's probabilities;
- For the transfer learning approach, all the layer's parameters from the convolutional base of the pre-trained model are extracted and fine-tuned, which yields the best performance for this specific dataset (see the results presented in Section 5.1);
- Following the work done by Gessert *et al.* [20] the Adam optimizer [33] is used, with $\rho_1 = 0.9$ and $\rho_2 = 0.999$;
- The convolutional base is frozen for the initial 2 epochs with a learning rate of 10^{-3}. The initial fine-tuning learning rate is 10^{-4} but is reduced by a factor of 10 when validation loss stops improving for 8 epochs. Each model trains for a maximum of 100 epochs but early stopping is performed whenever validation loss stops improving for 16 epochs;
- For each epoch, all the samples are shuffled before being feed into the network;
- Each batch is composed of 8 samples.
- Online and offline data augmentation is used, with random crops, rotations, and flips each with a probability of 0.5 (augmentation group 1). Offline data augmentation is used a class balancing measure and online data augmentation as a method to alleviate overfitting;
- For each training process, 3 models are saved: The model that obtained the highest BMA on the validation set, the model with the lowest loss on the validation set, and finally the resulting model from the last epoch. However, the performance on the test set is evaluated according to the model which attained the highest BMA on the validation set.

6.3.1 Softmax Threshold

Multi-class classification problems using deep neural networks often use a softmax layer at the last layer of the neural network classifier. The softmax probabilities of a softmax layer always add up to 1 and are spread across all classes with non-zero values. As such, the network will presumably try to optimize softmax probabilities by maximizing the softmax of the ground truth class and minimizing the probabilities for other classes. Even though the softmax predictions of an out-of-distribution sample in an 8-class model will add up to 1, one can make the assumption that the highest score will be lower than an in-distribution sample because the model will likely not be confident of its predictions.

Therefore, a threshold will be imposed on the softmax so that if the top softmax probability is lower than that threshold, then the sample will be labeled as "unknown", or more formally known as an out-of-distribution sample. However, there is no clear threshold value that will benefit the model the most. The experiments shown in Figure 6.15 suggest that low thresholds have close to no impact on the classification of the out-of-distribution samples, while high thresholds negatively impact the model performance.

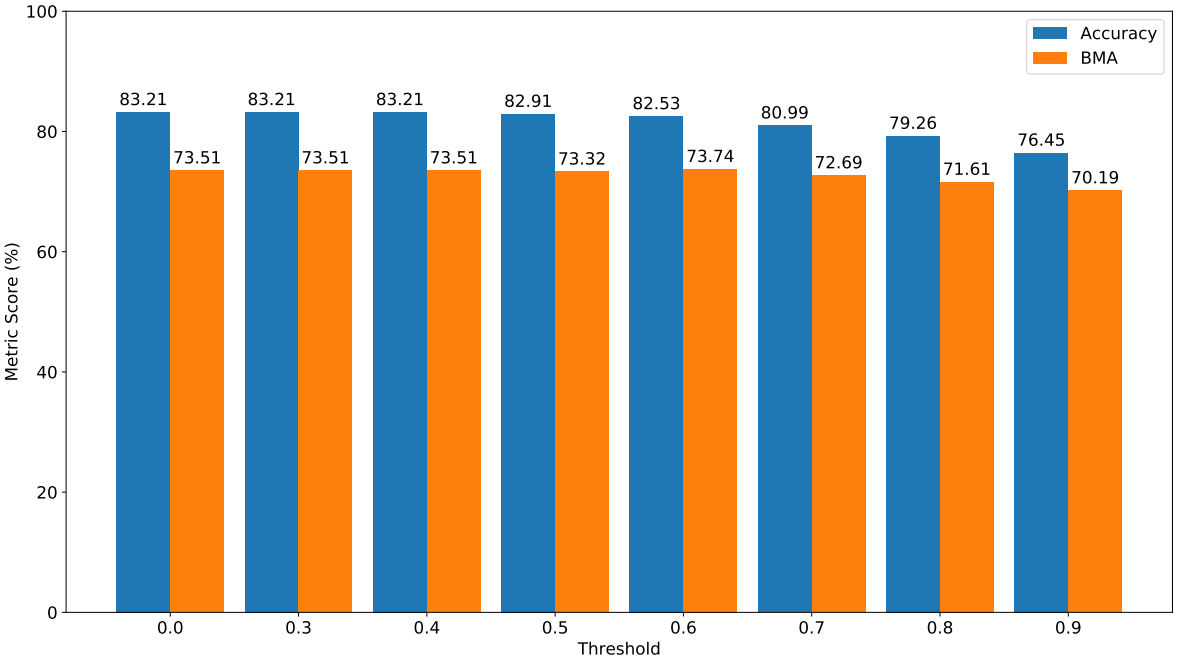

Figure 6.15: Comparison of different top-1 softmax threshold values for the detection of out of training distribution samples from the test set.

Considering the results, low threshold values having no impact stem from the fact that the DenseNet model often outputs high softmax scores towards a specific class (see the examples from Figure 5.1). This is an expected attribute of these types of deep learning models because they were optimized for 8 classes as opposed to 9, which means that they will try to maximize softmax scores towards one of the 8 classes rather than considering that a sample might not be part of the initial 8 class distribution of the training dataset.

This property also reflects in the results of higher thresholds, because such thresholds start negatively impacting correctly classified samples from other classes as seen by the confusion matrix in Figure 6.16. These results show a bright contrast with the ones obtained by Hendricks *et al.* [66] for his baseline approach towards out-of-distribution detection (recall Section 2.7). However, one should take into consideration that the in-distribution dataset used by them is far different from the out-of-distribution samples. More specifically, they presented results with MNIST [36] as an in-distribution dataset and out-of-distribution datasets such as Gaussian Noise, uniform noise and black and white CIFAR-10 [114] samples. Unfortunately, for the problem of skin lesion classification in- and out-of-distribution samples can be quite similar which makes this approach sub-optimal.

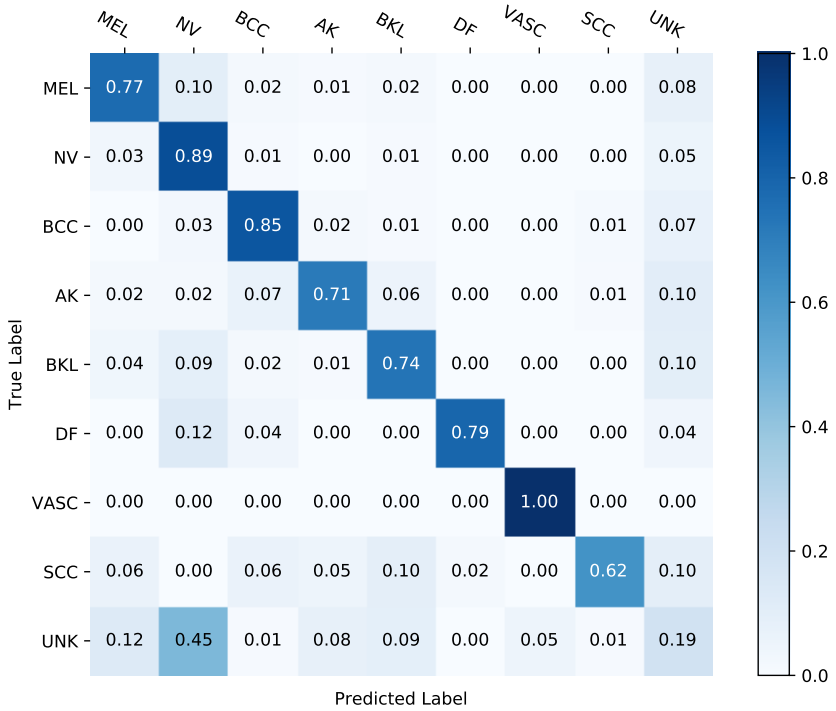

Figure 6.16: Confusion matrix of the predictions of the test set containing out of training distribution samples. The used model is the hyperparameter tuned DenseNet121 trained with 83432 class balanced samples from 8 different classes, and online data augmentation is performed with group 1 as a method to reduce overfitting. Out of distribution samples (*i.e.*, "unknown" samples) are identified using a top-1 softmax threshold of 0.7.

6.3.2 ODIN

As demonstrated in Section 2.7, there are 3 parameters within the ODIN that need tuning, namely: the temperature scaling T, the perturbation magnitude ε and the threshold σ. However, as this tuning procedure has already been done by Hsin-Wei Wang Wang [102] for the same dataset (ISIC 2019 challenge dataset), it has been decided to not tune these parameters, but to rather use the ones already tuned by this author. More specifically, $T = 2$, $\varepsilon = 0.0002$ and $\sigma = 0.90385$.

Results with this method present an accuracy of 69.08% and a BMA of 65.35% over the test set. As shown in the confusion matrix of Figure 6.17, it is clear that ODIN is not performing well enough as predictions towards the unknown class are evenly distributed across the 9 classes. It is hypothesized that ODIN is not a good option for out-of-distribution detection in skin lesion classification, because in and out of distribution samples are very similar to each other. This means that techniques like temperature scaling and adding small perturbations to the inputs have a small impact on the classification results.

These results contrast the results obtained by Liang *et al.* [67], where the in and the out-of-distribution samples were quite different from each other. More specifically, in distribution images belonged to the CIFAR-10 [114] which consists of 60000 images in 10 classes and the out-of-distribution samples were either from a uniform noise distribution, a Gaussian noise distribution, the Tiny ImageNet dataset (a subset of the ImageNet dataset [18]) which is

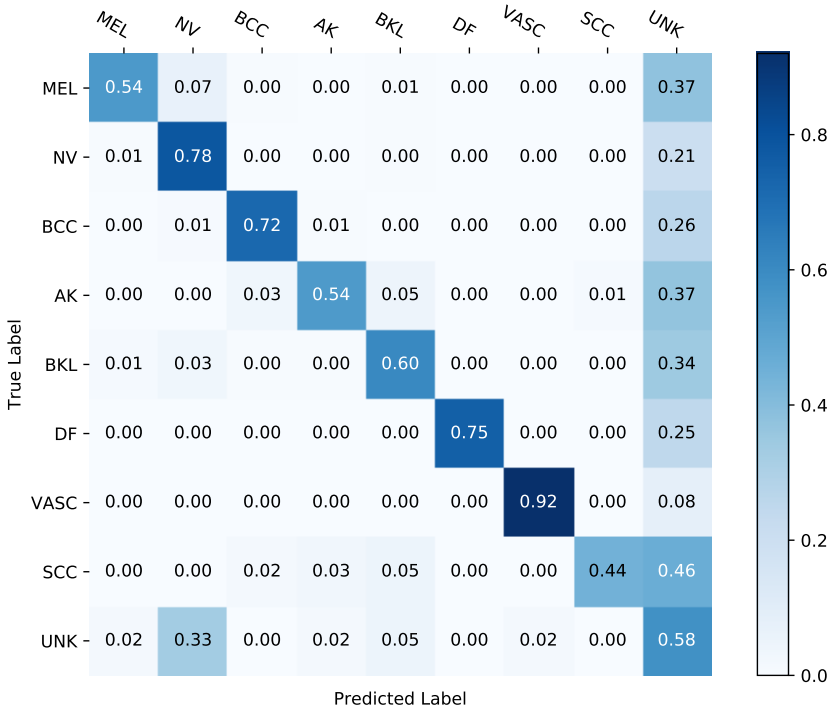

Figure 6.17: Confusion matrix of the predictions of the test set containing out of training distribution samples. The used model is the hyperparameter tuned DenseNet121 trained with 83432 class balanced samples from 8 different classes, and online data augmentation is performed with group 1 as a method to reduce overfitting. Out of distribution samples (*i.e.*, "unknown" samples) are identified using ODIN [67]

composed of 200 classes, or finally from the LSUN dataset [115] which is composed of 30 classes (10 scene categories and 20 object categories). From this, one can assume that ODIN only works well when the training dataset is far different from the out of distribution samples.

6.3.3 Outlier Class

From the results so far, it is evident that only using the softmax probabilities is not enough to determine whether a sample belongs to the training dataset. As such, the final approach towards out of distribution detection is to re-train the original DenseNet121 model with a 9th class with the out of distribution data described in Chapter 4.

Samples are split into train, validation, and test in a stratified way just like in the previous experiments (see Section 4.5). This means that very few samples are used to train, validate, and test this class. Therefore, to alleviate this problem, the outlier class is oversampled through the data augmentation group 1 so that all classes are balanced (just like the final approach defined in Section 4.2). In practice, this means that a DenseNet121 is trained with 93861 samples (most of them synthetic).

As shown in the confusion matrix in Figure 6.18, results are considerably better than the previous approaches with an overall accuracy of 84.72% and a BMA of 74.98%. However, it is still very apparent that the "unknown" class has significantly lower performance than the rest of the classes. These findings can be attributed to the small amount of rather heterogeneous samples in this class used to train the model. One can also observe that there are a lot of

Figure 6.18: Confusion matrix of the test set predictions on the hyperparameter tuned DenseNet121 trained with 93861 samples class balanced samples from 9 different classes. The 9th class is composed of the out of distribution data described in Chapter 4. Online data augmentation is performed with the group 1 as a method to reduce overfitting.

out-of-distribution samples (UNK) that are being classified as Melanocytic Nevus (NV) and then again one can speculate that this might be due to a lack of data for the "unknown" class.

6.3.4 Approach Comparison

Finally, these 3 approaches are compared in Table 6.2, with respect to the test BMA and accuracy scores. It is evident from the results that training an outlier class outperforms both softmax thresholding and the ODIN method. These findings corroborate the results attained in the literature, specifically, Hsin-Wei Wang reported that the ODIN classifier performed poorly due to an insignificant difference between in and out of distribution samples [102]. In contrast, approaches such as the one done by Gessert *et al.* [21] attained one of the best results in 9-class classification, presumably because of the availability of a big in-house dataset used to train an outlier class composed of out of distribution samples (in-distribution being the original 8 classes). As such, more effort should be put into gathering a bigger set of "unknown" samples.

Out-of-distribution method	BMA	Accuracy
Softmax threshold (0.7)	0.721	0.823
ODIN	0.654	0.691
Outlier class	0.750	0.847

Table 6.2: Comparison of BMA and accuracy scores on the test set for the different methodologies to detect out of training distribution samples.

However, it is important to point out some limitations of the outlier class approach. Specifically, there are very few "unknown" samples within the test set, which might be an indicator that the results might be biased. Furthermore, a good portion of the samples from the "unknown" test set belong to "Lentigo NOS". This might also be negatively impacting the training process for the outlier class approach because the most effective way of minimizing the loss within the "unknown" class is by correctly classifying "Lentigo NOS" samples. Moreover, the test set will have more "Lentigo NOS" samples than the rest of the classes which can also be biasing the test set results.

6.4 Results Discussion

Finally, one can now compare the different methods employed throughout this work in Table 6.3. Planned experiments in Section 6.1 corroborated the importance of dataset size for deep neural networks, which significantly improved the performance of the model in comparison to the DenseNet201 employed in Chapter 5. However, sufficient labeled data for skin lesions is hard to come by, which highlights the importance of data augmentation. The presented comparisons show that simple data augmentation techniques can significantly improve generalization performance through offline data augmentation as a class balancing measure. Furthermore, while online data augmentation proved to be an excellent approach to reduce overfitting, offline data augmentation has a big impact on the generalization performance of underrepresented classes.

Approach Step	BMA	Accuracy
Pre-trained model choice (see Section 5.1)	≈ 0.579	≈ 0.746
Hyperparameter tuned model (see Section 5.2)	≈ 0.585	≈ 0.754
Full dataset model without online DA (see Section 6.1.2)	≈ 0.636	≈ 0.799
Full dataset model with online DA (see Section 6.1.2)	≈ 0.769	≈ 0.849
Class balanced model (see Section 6.1.3)	≈ 0.815	≈ 0.867
Ensemble of 3 models (see Section 6.2)	≈ 0.846	≈ 0.891
Model with out-of-distribution detection (see Section 6.3)	≈ 0.750	≈ 0.847

Table 6.3: BMA and accuracy on the test set for the different models created throughout this work.

This study confirms that ensemble learning can further improve the results by using different CNN architectures. Experimentation also underlines the importance of choosing the best models for the ensemble, otherwise, it will result in a decrease in performance as a consequence of the averaging scheme. However, one should attain whether it is worth to make such an approach for a practical application of these classifiers. For example, if one were to use the best model ensemble, namely, the 3 model ensemble with DenseNet121, InceptionResNetV2, and EfficientNetB2, then one could expect a performance increase of 3% (see Table 6.3), but it would require approximately 3 times as much time to predict the class label for all the models, which might become a limiting factor when deploying this classifier into a production environment. Therefore, such an approach is useful for benchmark challenges, but it could be impractical for real-world usage.

The presented approach achieves competitive results in the ISIC 2019 challenge without the need for additional external data in the original 8 classes. It outperforms the state-of-the-art approaches for 8-class classification on both single-model performance and multi-model performance through ensembling (see Table 6.4). Results from 9-class classification show a significant decrease in both BMA and accuracy scores in comparison with 8-class classification due to the underwhelming results of the out-of-distribution detection.

Approach	In distribution (8 Classes)		Out of distribution (9 Classes)	
	BMA_{test}	A_{test}	BMA_{test}	A_{test}
3 Model Ensemble	0.846	0.891	0.775	0.858
DenseNet121	0.815	0.867	0.750	0.847
Gessert *et al.* [21]	0.725	NA	0.636	NA
Zhou *et al.* [25]	0.753	NA	0.607	NA
Hsin-Wei Wang [102]	0.828	NA	0.505	NA

Table 6.4: Approach comparison with state-of-the-art approaches on 8- and 9-class classification for the ISIC 2019. The proposed approaches are highlighted in bold, namely the "3 Model Ensemble" and the "DenseNet121". The in distribution models (8 classes) are trained with 83432 class balanced samples and the out of distribution models (9 classes) are trained with 93861 class balanced samples, where the 9th class is composed of samples from a different distribution of the original 8 (see subsection 4.2.3). The approaches from Gessert *et al.* [21], Zhou *et al.*, and Hsin-Wei Wang [102] are ensembles because it is their best performing models. For a detailed single-model performance of these approaches, relate to their original papers.

The obtained results for 8-class classification of skin lesions in the ISIC 2019 dataset, resonate the extensive experimentation done throughout this work, that leads to a systematically better performing model step-by-step. However, some limitations are still present, more specifically, our approach for the "unknown" class was unable to attain good generalization performance, with the highest performing method being the outlier class. This is a fairly limited approach in the sense that one does not know the real-world distribution of this class. Even though other strategies such as softmax thresholding and the ODIN were experimented with, those methods proved to be very ineffective in skin lesion diagnosis.

Even though the presented results are superior to the state-of-the-art (Gessert *et al.* [21]) on 9-class classification, it should be noted that the test set used to compute the BMA and accuracy for the 9-class classification in this approach was part of the unknown dataset gathered from the ISIC archive (see subsection 4.2.3), rather than the 9-class test set originally used to compare different approaches on ISIC 2019. This is the case because the ISIC organization does not provide the ground truth data related to the ISIC 2019 challenge test data. Furthermore, they do not allow any new submissions towards this challenge leaderboard, in order to compute this approach's metrics.

CHAPTER 7

Conclusion

This dissertation considered the development of a multi-class dermoscopic image classifier for the diagnosis of 8 types of skin lesions based on pre-trained CNNs. First, pre-trained network architectures were comparatively evaluated and their hyperparameters optimized. Second, several efforts have been made to improve the generalization capability of the deep network. In this context, the impact of data augmentation, class imbalance, and ensembling techniques was demonstrated. Globally, the approach taken throughout the study for skin lesion classification provides significant improvements over current state-of-the-art results from the ISIC 2019 challenge. Nevertheless, the experiments performed with the "unknown" class highlight the need for a better understanding of current limitations of out of training distribution detection methods. Taking this into considerations, this chapter presents the final remarks and underlines points for future research.

7.1 Discussion

In conclusion, the following remarks can be drawn from the results obtained in Chapter 5 and Chapter 6:

- As the dataset of the ISIC 2019 dataset is far different from ImageNet, extracting and fine-tuning all the parameters up to the highest layer consistently yielded better performance on all pre-trained models when compared with only training the classifier. Therefore, when a pre-trained model is optimized for a far different dataset, the fine-tuning process will allow it to generalize knowledge to a different domain;
- In skin lesion classification, by using a transfer learning approach, the choice of the pre-trained model's architecture can have a substantial impact on the generalization performance of the model. Furthermore, more recent model architectures proved to bring major improvements in comparison with older architectures. More specifically, EfficientNets and DenseNets provide good results with a low amount of trainable parameters and scale well with the available computational resources. However, architectures

like VGG perform poorly and do not scale well with an increase in the number of layers and trainable parameters;

- Transfer learning provides a good framework for the classification of skin lesions as most pre-trained models are well optimized for a wide range of hyperparameters. This property removes the need for an extensive hyperparameter optimization often required on learning from scratch approaches, which allows beginners to easily train and test deep learning models without machine learning expertise;
- The data augmentation techniques used to oversample the original dataset should be carefully hand-picked depending on the classification problem. Specifically, in skin lesion diagnosis, augmentation techniques that change pixel intensities, distort the original image, or add noise into the image proved to perform worse than simpler techniques like rotations or crops, because they remove or alter important information about the lesions;
- Online data augmentation emerges as a method to significantly reduce overfitting in the problem of skin lesion classification. While simple augmentation techniques like flips, crops, and rotations are a good baseline for applying online data augmentation, complex augmentation techniques like perspective transformations have a much bigger impact on overfitting reduction;
- Ensembling multiple models trained with different CNN architectures can slightly improve the overall performance, which can be useful for challenges such as ISIC 2019 to gain an edge over other approaches. However, in a real-world scenario, a faster and more practical approach is much more preferable than a slight performance increase. For example, one could use a single small model like in the DenseNet121 and attain similar results;
- Presumably, out-of-distribution detection through softmax thresholding or ODIN performs badly in the context of classifying skin lesions, because the samples from the unknown dataset are not substantially different from the original 8 classes from the ISIC 2019 challenge. However, re-training the model with an outlier class can become a viable approach as long as more effort is put into creating a broad distribution of samples from multiple lesions outside of the main 8 from the original ISIC 2019 training dataset.

7.2 Limitations and Future Work

Several limitations remain present concerning this approach towards ISIC 2019, and, generally, for multi-class deep learning based skin lesion classifiers. However, such limitations provide exciting new research opportunities that should be explored in the future, more specifically:

- There are still limitations to overcome concerning the benchmark challenges presented by the ISIC committee. Namely, there is a considerable lack of samples of less common skin lesions in the ISIC 2019 dataset. Therefore, gathering more samples towards these types of classes should be a focal point to improve the performance of deep learning based

models. Furthermore, in the ISIC 2019, performance metrics for 8-class classification (in-distribution) are not discriminated in the website's leaderboard. Finally, the ISIC 2019 test set ground truth is not provided to the public after the challenges have ended. This presents some limitations to independent authors who wish to directly compare results to previous submissions and should be addressed in future versions of this challenge;

- As described in Chapter 4, the test set used is fairly small, highly imbalanced, and might not be representative of real-world skin lesions. For example, the ISIC 2019 dataset is mostly composed of samples from a very narrow skin pigmentation interval [24], which might be impactful on the diagnosis of lesions with other skin tones. Methods such as test time data augmentation were not addressed in this work and could ultimately help reduce the footprint of this issue in real-world test sets. However, it would be interesting to assess the performance of these models on an unfiltered test set provided by a local hospital or clinic;
- As some of the results related to online data augmentation were made in later developments of this approach, they were left aside from experiments with class balancing, ensembling, and out of training distribution detection. Therefore, one could further explore different augmentation groups and their impact on the overfitting reduction on other datasets. For example, one could evaluate this method on the upcoming ISIC 2020 challenge dataset, which tackles the problem of melanoma detection by using multiple samples for a single patient [116];
- Training an outlier class is a limited approach as there is no clearly defined distribution of the "unknown" samples for the ISIC 2019 test set. As such, a survey of the distribution of a real-world test set would be useful. This would allow for a more concise out of distribution set of data to be organized, which would further improve the performance of the "unknown" class and presumably allow this model to be viable for deployment into a production environment;
- One property of this approach towards automated skin lesion classification was left aside, namely, the interpretability of the model. As it has been discussed in Section 3.4, these deep learning models can be seen as a black box because they do not provide arguments so that one could interpret their decisions. A possible solution for this problem is through the incorporation of medical knowledge within the approach to make a hierarchical classifier (*e.g.*, Barata *et al.* [107]) or make use of the kernel filters during the inference phase (*e.g*, Van Mole *et al.* [106]);
- Although challenges like ISIC are great for pushing the performance of skin lesion classifiers forward, they do not address problems related to the deployment of these models into real-world scenarios. Integrating these models into an easy-to-use tool and with a user-friendly interface, will ultimately benefit both dermatologists and patients as it abstracts unnecessary machine learning related concerns. As such, an important research focus for the future is the deployment of these deep learning based models into the clinical workflow of a skin professional.

7.3 Contributions

The following are the contributions from this work towards the research community:

- Santos, F., Silva, F., & Georgieva, P. (2020). Automated Diagnosis of Skin Lesions. 2020 IEEE 10th International Conference on Intelligent Systems (IS), August 28-30 (PAPER ACCEPTED).
- Santos, Fábio, Skin lesion diagnosis using a multi-class deep learning classifier, (2020), GitHub repository, `https://github.com/FabioTomaz/msc`.

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] H. W. Rogers, M. A. Weinstock, S. R. Feldman, and B. M. Coldiron, "Incidence estimate of nonmelanoma skin cancer (keratinocyte carcinomas) in the us population, 2012", *JAMA Dermatology*, vol. 151, no. 10, pp. 1081–1086, Oct. 2015, ISSN: 21686068. DOI: 10.1001/jamadermatol.2015.1187.

[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.

[4] 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.

[5] 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.

[6] G. Argenziano and H. P. Soyer, *Dermoscopy of pigmented skin lesions - a valuable tool for early diagnosis of melanoma*, Jul. 2001. DOI: 10.1016/S1470-2045(00)00422-8.

[7] H. Kittler, H. Pehamberger, K. Wolff, and M. Binder, *Diagnostic accuracy of dermoscopy*, Mar. 2002. DOI: 10.1016/S1470-2045(02)00679-4.

[8] E. Okur and M. Turkan, "A survey on automated melanoma detection", *Engineering Applications of Artificial Intelligence*, vol. 73, pp. 50–67, Aug. 2018, ISSN: 09521976. DOI: 10.1016/j.engappai.2018.04.028.

[9] 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.

[10] N. K. Mishra and M. E. Celebi, "An overview of melanoma detection in dermoscopy images using image processing and machine learning", Jan. 2016. arXiv: 1601.07843.

[11] J. S. Henning, S. W. Dusza, S. Q. Wang, A. A. Marghoob, H. S. Rabinovitz, D. Polsky, and A. W. Kopf, "The cash (color, architecture, symmetry, and homogeneity) algorithm for dermoscopy", *Journal of the American Academy of Dermatology*, vol. 56, no. 1, pp. 45–52, Jan. 2007, ISSN: 01909622. DOI: 10.1016/j.jaad.2006.09.003.

[12] F. Nachbar, W. Stolz, T. Merkle, A. B. Cognetta, T. Vogt, M. Landthaler, P. Bilek, O. Braun-Falco, and G. Plewig, "The abcd rule of dermatoscopy: High prospective value in the diagnosis of doubtful melanocytic skin lesions", *Journal of the American Academy of Dermatology*, vol. 30, no. 4, pp. 551–559, Apr. 1994, ISSN: 01909622. DOI: 10.1016/S0190-9622(94)70061-3.

[13] P. Tschandl, C. Rosendahl, B. N. Akay, G. Argenziano, A. Blum, R. P. Braun, H. Cabo, J. Y. Gourhant, J. Kreusch, A. Lallas, J. Lapins, A. Marghoob, S. Menzies, N. M. Neuber, J. Paoli, H. S. Rabinovitz, C. Rinner, A. Scope, H. P. Soyer, C. Sinz, L. Thomas, I. Zalaudek, and H. Kittler, "Expert-level diagnosis of nonpigmented skin cancer by combined convolutional neural networks", *JAMA Dermatology*, vol. 155, no. 1, pp. 58–65, Jan. 2019, ISSN: 21686068. DOI: 10.1001/jamadermatol.2018.4378.

[14] 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.

[15] 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.

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

[17] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P. M. Agapow, M. Zietz, M. M. Hoffman, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. E. Carpenter, A. Shrikumar, J. Xu, E. M. Cofer, C. A. Lavender, S. C. Turaga, A. M. Alexandari, Z. Lu, D. J. Harris, D. Decaprio, Y. Qi, A. Kundaje, Y. Peng, L. K. Wiley, M. H. Segler, S. M. Boca, S. J. Swamidass, A. Huang, A. Gitter, and C. S. Greene, "Opportunities and obstacles for deep learning in biology and medicine", *Journal of the Royal Society Interface*, vol. 15, no. 141, 2018, ISSN: 17425662. DOI: 10.1098/rsif.2017.0387.

[18] 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.

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?", *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3320–3328, Nov. 2014. arXiv: 1411.1792.

[20] 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.

[21] N. Gessert, M. Nielsen, M. Shaikh, R. Werner, and A. Schlaefer, "Skin lesion classification using ensembles of multi-resolution efficientnets with meta data", *MethodsX*, vol. 7, Oct. 2019. arXiv: 1910.03910.

[22] F. Perez, C. Vasconcelos, S. Avila, and E. Valle, "Data augmentation for skin lesion analysis", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11041 LNCS, pp. 303–311, Sep. 2018. DOI: 10.1007/978-3-030-01201-4_33. arXiv: 1809.01442.

[23] H. He and E. A. Garcia, "Learning from imbalanced data", *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, ISSN: 10414347. DOI: 10.1109/TKDE.2008.239.

[24] *International skin imaging collaboration (isic) challenge 2019*, 2019. [Online]. Available: https://challenge2019.isic-archive.com/ (visited on 11/23/2019).

[25] S. Zhou, Y. Zhuang, and R. Meng, "Multi-category skin lesion diagnosis using dermoscopy images and deep cnn ensembles", DysionAI, Tech. Rep., 2019.

[26] F. Pollastri, J. Maroñas, M. Parreño, F. Bolelli, R. Paredes, C. Grana, and A. Albiol, "Aimagelab-prhlt at isic challenge 2019", AImageLab, Tech. Rep., 2019.

[27] 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.

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

[29] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, ISSN: 09324194. DOI: 10.1007/BF02551274.

[30] 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.

[31] N. Qian, "On the momentum term in gradient descent learning algorithms.", *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[32] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[33] 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*, ICLR, Dec. 2015. arXiv: 1412.6980.

[34] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", Tech. Rep., 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", *Proceedings of the ieee international conference on computer vision*, pp. 1026–1034, 2015. arXiv: 1502.01852.

[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, ISSN: 00189219. DOI: 10.1109/5.726791.

[37] Y. LeCun, "Generalization and network design strategies", *Connectionism in perspective*, vol. 19, 1989.

[38] *Cs231n convolutional neural networks for visual recognition*. [Online]. Available: https://cs231n.github.io/convolutional-networks/ (visited on 03/15/2020).

[39] 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.

[40] 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.

[41] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, *A state-of-the-art survey on deep learning theory and architectures*, Mar. 2019. DOI: 10.3390/electronics8030292.

[42] 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.

[43] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks", Aug. 2016. arXiv: 1608.06993.

[44] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks", May 2019. arXiv: 1905.11946.

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, IEEE Computer Society, Oct. 2015, pp. 1–9, ISBN: 9781467369640. DOI: 10.1109/CVPR.2015.7298594. arXiv: 1409.4842.

[46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, IEEE Computer Society, Dec. 2016, pp. 2818–2826, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.308. arXiv: 1512.00567.

[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, International Machine Learning Society (IMLS), 2015, pp. 448–456, ISBN: 9781510810587. arXiv: 1502.03167.

[48] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning", in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, AAAI press, Feb. 2017, pp. 4278–4284. arXiv: 1602.07261.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9908 LNCS, pp. 630–645, Mar. 2016. arXiv: 1603.05027.

[50] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges", *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, May 2017, ISSN: 14774054. DOI: 10.1093/bib/bbx044.

[51] T. G. Dipanjan Sarkar, Raghav Bali, *Hands-On Transfer Learning with Python*, First. Packt Publishing, 2018, ISBN: 978-1-78883-130-7.

[52] 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).

[53] M. Lin, Q. Chen, and S. Yan, "Network in network", *arXiv preprint*, p. 10, Dec. 2013. arXiv: 1312.4400.

[54] Y. Tang, "Deep learning using linear support vector machines", Jun. 2013. arXiv: 1306.0239.

[55] J. Grus, *Data Science from Scratch: First Principles with Python*. Beijing: O'Reilly, 2015, ISBN: 978-1-4919-0142-7. [Online]. Available: http://my.safaribooksonline.com/97814919-01427.

[56] A. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance", *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004. DOI: 10.1145/1015330.1015435.

[57] 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.

[58] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning", *Journal of Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, ISSN: 21961115. DOI: 10.1186/s40537-019-0197-0.

[59] 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).

[60] 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*, ICLR, Apr. 2019. arXiv: 1704.00109.

[61] J. Xie, B. Xu, and Z. Chuang, "Horizontal and vertical ensemble with deep representation for classification", Jun. 2013. arXiv: 1306.2759.

[62] 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.

[63] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization", *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, vol. 2, pp. 876–885, Mar. 2018. arXiv: 1803.05407.

[64] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, ICLR, Dec. 2015. arXiv: 1412.6572.

[65] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety", Jun. 2016. arXiv: 1606.06565.

[66] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks", in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, ICLR, Oct. 2016. arXiv: 1610.02136.

[67] 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.

[68] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network", Mar. 2015. arXiv: 1503.02531.

[69] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks", *34th International Conference on Machine Learning, ICML 2017*, vol. 3, pp. 2130–2143, Jun. 2017. arXiv: 1706.04599.

[70] 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.

[71] A. P. Kassianos, J. D. Emery, P. Murchie, and F. M. Walter, "Smartphone applications for melanoma detection by community, patient and generalist clinician users: A review", *British Journal of Dermatology*, vol. 172, no. 6, pp. 1507–1518, Jun. 2015, ISSN: 13652133. DOI: 10.1111/bjd.13665.

[72] A. A. Zaidan, B. B. Zaidan, O. S. Albahri, M. A. Alsalem, A. S. Albahri, Q. M. Yas, and M. Hashim, *A review on smartphone skin cancer diagnosis apps in evaluation and benchmarking: Coherent taxonomy, open issues and recommendation pathway solution*, Sep. 2018. DOI: 10.1007/s12553-018-0223-9.

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

[74] T. Maier, D. Kulichova, K. Schotten, R. Astrid, T. Ruzicka, C. Berking, and A. Udrea, "Accuracy of a smartphone application using fractal image analysis of pigmented moles compared to clinical diagnosis and histological result", *Journal of the European Academy of Dermatology and Venereology*, vol. 29, no. 4, pp. 663–667, Apr. 2015, ISSN: 14683083. DOI: 10.1111/jdv.12648.

[75] *Federal trade commission cracks down on marketers of melanoma detection apps*, Feb. 2015. [Online]. Available: https://www.ftc.gov/news-events/press-releases/2015/02/ftc-cracks-down-marketers-melanoma-detection-apps (visited on 07/02/2020).

[76] M. Emre Celebi, Q. Wen, S. Hwang, H. Iyatomi, and G. Schaefer, "Lesion border detection in dermoscopy images using ensembles of thresholding methods", *Skin Research and Technology*, vol. 19, no. 1, Feb. 2013, ISSN: 0909752X. DOI: 10.1111/j.1600-0846.2012.00636.x. arXiv: 1312.7345.

[77] Q. Abbas, M. E. Celebi, I. Fondón García, and M. Rashid, "Lesion border detection in dermoscopy images using dynamic programming", *Skin Research and Technology*, vol. 17, no. 1, pp. 91–100, Feb. 2011, ISSN: 0909752X. DOI: 10.1111/j.1600-0846.2010.00472.x.

[78] M. Mete, S. Kockara, and K. Aydin, "Fast density-based lesion detection in dermoscopy images", *Computerized Medical Imaging and Graphics*, vol. 35, no. 2, pp. 128–136, Mar. 2011, ISSN: 08956111. DOI: 10.1016/j.compmedimag.2010.07.007.

[79] H. Zhou, X. Li, G. Schaefer, M. E. Celebi, and P. Miller, "Mean shift based gradient vector flow for image segmentation", *Computer Vision and Image Understanding*, vol. 117, no. 9, pp. 1004–1016, Sep. 2013, ISSN: 1090235X. DOI: 10.1016/j.cviu.2012.11.015.

[80] X. Yuan, N. Situ, and G. Zouridakis, "A narrow band graph partitioning method for skin lesion segmentation", *Pattern Recognition*, vol. 42, no. 6, pp. 1017–1028, Jun. 2009, ISSN: 00313203. DOI: 10.1016/j.patcog.2008.09.006.

[81] A. Wong, J. Scharcanski, and P. Fieguth, "Automatic skin lesion segmentation via iterative stochastic region merging", *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 6, pp. 929–936, Nov. 2011, ISSN: 10897771. DOI: 10.1109/TITB.2011.2157829.

[82] 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.

[83] H. Pehamberger, A. Steiner, and K. Wolff, "In vivo epiluminescence microscopy of pigmented skin lesions. i. pattern analysis of pigmented skin lesions", *Journal of the American Academy of Dermatology*, vol. 17, no. 4, pp. 571–583, Oct. 1987, ISSN: 01909622. DOI: 10.1016/S0190-9622(87)70239-4.

[84] M. Barzegari, H. Ghaninezhad, P. Mansoori, A. Taheri, Z. S. Naraghi, and M. Asgari, "Computer-aided dermoscopy for diagnosis of melanoma", *BMC Dermatology*, vol. 5, no. 1, p. 8, Jul. 2005, ISSN: 14715945. DOI: 10.1186/1471-5945-5-8.

[85] R. H. Johr, "Dermoscopy: Alternative melanocytic algorithms - the abcd rule of dermatoscopy, menzies scoring method, and 7-point checklist", *Clinics in Dermatology*, vol. 20, no. 3, pp. 240–247, 2002, ISSN: 0738081X. DOI: 10.1016/S0738-081X(02)00236-5.

[86] E. Unlu, B. N. Akay, and C. Erdem, "Comparison of dermatoscopic diagnostic algorithms based on calculation: The abcd rule of dermatoscopy, the seven-point checklist, the three-point checklist and the cash algorithm in dermatoscopic evaluation of melanocytic lesions", *Journal of Dermatology*, vol. 41, no. 7, pp. 598–603, 2014, ISSN: 13468138. DOI: 10.1111/1346-8138.12491.

[87] F. M. Walter, A. T. Prevost, J. Vasconcelos, P. N. Hall, N. P. Burrows, H. C. Morris, A. L. Kinmonth, and J. D. Emery, "Using the 7-point checklist as a diagnostic aid for pigmented skin lesions in general practice: A diagnostic validation study", *British Journal of General Practice*, vol. 63, no. 610, 2013, ISSN: 09601643. DOI: 10.3399/bjgp13X667213.

[88] Z. Hu, J. Tang, Z. Wang, K. Zhang, L. Zhang, and Q. Sun, "Deep learning for image-based cancer detection and diagnosis - a survey", *Pattern Recognition*, vol. 83, pp. 134–149, Nov. 2018, ISSN: 00313203. DOI: 10.1016/j.patcog.2018.05.014.

[89] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, *A survey on deep learning in medical image analysis*, Dec. 2017. DOI: 10.1016/j.media.2017.07.005.

[90] 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.

[91] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, and G. Research, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", Tech. Rep. arXiv: 1603.04467v2.

[92] 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.

[93] *The international skin imaging collaboration: Melanoma project.* [Online]. Available: https://www.isic-archive.com/%7B%5C#%7D (visited on 07/08/2020).

[94] *International skin imaging collaboration (isic) challenge 2018 task 3: Lesion diagnosis*, 2018. [Online]. Available: https://challenge2018.isic-archive.com/task3/ (visited on 11/23/2019).

[95] A. Nozdryn-Plotnicki, J. Yap, and W. Yolland, *Ensembling convolutional neural networks for skin cancer classification*, 2018.

[96] E. Finlayson, Graham and Trezzi, "Shades of gray and colour constancy", *Proceedings of the 12th Color Imaging Conference*, pp. 37–41, 2004.

[97] 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.

[98] 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.

[99] J. Kawahara, S. Daneshvar, G. Argenziano, and G. Hamarneh, "Seven-point checklist and skin lesion classification using multitask multimodal neural nets", *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 2, pp. 538–546, Mar. 2019, ISSN: 2168-2194. DOI: 10.1109/JBHI.2018.2824327.

[100] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout", Aug. 2017. arXiv: 1708.04552.

[101] 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.

[102] H.-W. Wang, "Skin lesion classification using ensemble of convolutional neural networks and out-of-distribution detector", Tech. Rep., 2019. [Online]. Available: https://github.com/wanghsinwei/isic-2019.

[103] L. Yu, H. Chen, Q. Dou, J. Qin, and P. A. Heng, *Automated melanoma recognition in dermoscopy images via very deep residual networks*, Apr. 2017. DOI: 10.1109/TMI.2016.2642839.

[104] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, International Machine Learning Society (IMLS), Feb. 2015, pp. 448–456, ISBN: 9781510810587. arXiv: 1502.03167.

[105] P. Ly, D. Bein, and A. Verma, *New compact deep learning model for skin cancer recognition*, Aug. 2019. DOI: 10.1109/uemcon.2018.8796628.

[106] P. Van Molle, M. De Strooper, T. Verbelen, B. Vankeirsbilck, P. Simoens, and B. Dhoedt, "Visualizing convolutional neural networks to improve decision support for skin lesion classification", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11038 LNCS, Springer Verlag, Sep. 2018, pp. 115–123, ISBN: 9783030026271. DOI: 10.1007/978-3-030-02628-8_13. arXiv: 1809.03851.

[107] C. Barata, M. E. Celebi, and J. S. Marques, "Explainable skin lesion diagnosis using taxonomies", *Pattern Recognition*, p. 107 413, May 2020, ISSN: 00313203. DOI: 10.1016/j.patcog.2020.107413.

[108] M. E. Celebi, N. Codella, and A. Halpern, *Dermoscopy image analysis: Overview and future directions*, Mar. 2019. DOI: 10.1109/JBHI.2019.2895803.

[109] J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann, "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study", *PLOS Medicine*, vol. 15, no. 11, A. Sheikh, Ed., e1002683, Nov. 2018, ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1002683.

[110] 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.

[111] 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.

[112] 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.

[113] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization", Oct. 2017. arXiv: 1710.09412.

[114] A. Krizhevsky, "Learning multiple layers of features from tiny images", Tech. Rep., Apr. 2009. [Online]. Available: http://www.cs.toronto.edu/%7B~%7Dkriz/cifar.html.

[115] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop", Jun. 2015. arXiv: 1506.03365.

[116] *The international skin imaging collaboration (isic) 2020 challenge dataset*, 2020. DOI: https://doi.org/10.34970/2020-ds01. (visited on 06/09/2020).