TÉCNICO LISBOA

Deep Hierarchical Diagnosis of Skin Lesions

Rúben dos Santos Gomes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Ana Catarina Fidalgo Barata
Prof. Jorge dos Santos Salvador Marques

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Ana Catarina Fidalgo Barata
Member of the Committee: Prof. Jacinto Carlos Marques Peixoto do Nascimento

July 2020

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all, I would like to thank my parents and brother for providing me the means to accomplish my academic education.

I would like to thank my supervisors, Prof. Jorge Marques and Dr. Catarina Barata, for their guidance, availability, and assertive remarks during the execution of this work. Also, a thanks for providing me the physical resources, which allowed me to execute the work.

I would like to thank Inês Neves, who followed me closely during these 5 years, giving me strength and courage to always pursue the best academic results. Also for having the patience to listen my enthusiastic academic thoughts.

Last but not least, I would like to thank my friends and remainder family for their support over the last 5 years.

Abstract

The early detection of skin cancer is very important because it increases significantly the patient's prognosis, and techniques of data classification and pattern recognition are important tools to accomplish it.

Skin lesions can be organized in a hierarchical structure, where in the first level the melanocytic lesions and non-melanocytic lesions are splitted, and in each of these sub-groups the benign and malignant lesions are further discriminated. However, this knowledge has been disregarded by most automatic methods. Thus, in this thesis we propose to investigate this issue.

In this work, we implement and compare the results of two different models based in deep learning, to diagnose skin lesion's images: (i) a flat model, where the inference is done in a single decision which involves all the categories, and (ii) a hierarchical model that explores the hierarchical structure of skin lesions' organization, where the inference for each example involves more than one decision. Afterwards, a mixed model, which combines these approaches, was implemented in order to capture the strengths of each one.

With the analysis of the models' results we verified that the hierarchical model is affected by the error propagation of the intermediate decisions, in comparison with the flat classifier. However, the mixed model allows to improve significantly the results over both approaches.

Additionally, since it is very important to obtain good results even when a small amount of data is available, we studied the effects of applying the hierarchical model in a dataset with reduced size. We verified that in this case, the hierarchical model's results are significantly worse, suggesting that the

hierarchical model performs better when applied in datasets of bigger size.

Keywords

Deep Learning, Hierarchical Classifier, Skin Lesions.

Resumo

A deteção precoce de cancros de pele é muito importante porque aumenta significativamente a probabilidade de sobrevivência do paciente. Neste contexto, técnicas de classificação de dados e reconhecimento de padrões são ferramentas importantes para o conseguir.

As lesões de pele podem ser organizadas numa estrutura hierárquica, em que no primeiro nível se dividem as lesões melanocíticas das não melanocíticas, e em cada um destes sub-grupos se dividem em benignas ou malignas. Contudo, este conhecimento tem sido desaproveitado na maioria dos métodos automáticos. Por esse motivo, nesta tese investigamos o resultado de utilizar este conhecimento.

Neste trabalho, estudaram-se dois tipos de modelos, baseados em aprendizagem profunda, para classificar imagens de lesões de pele: (i) um modelo tradicional, em que a inferência é realizada numa única decisão que envolve todas as categorias, e (ii) um modelo hierárquico que explora a estrutura hierárquica da organização de lesões de pele, em que a inferência é realizada através de várias decisões. Posteriormente, estudou-se um modelo misto que combina estas duas abordagens, de forma a capturar os pontos fortes de cada uma.

Com a análise dos resultados dos modelos propostos verificou-se que o modelo hierárquico é afetado pela propagação de erros das decisões intermédias, em relação ao classificador tradicional. No entanto, o modelo misto permite melhorar significativamente os resultados de ambas as abordagens.

Adicionalmente, dada a importância de conseguir obter bons resultados, mesmo quando a quantidade de dados disponível é reduzida, estudou-se o efeito da aplicação do modelo hierárquico num conjunto de dados de reduzida dimensão. Verificou-se que neste caso os resultados do modelo hierárquico em relação ao tradicional são significativamente piores, sugerindo que o modelo hierárquico tem uma

maior vantagem quando aplicado em conjuntos de dados de maior dimensão.

Palavras Chave

Aprendizagem Profunda, Classificador Hierárquico, Lesões de Pele.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

Adam	Adaptive Moment Estimation
AKIEC	Actinic Keratosis
ANN	Artificial Neural Network
ANNs	Artificial Neural Networks
BACC	Balanced Accuracy
BCC	Basal Cell Carcinoma
BKL	Benign Keratosis
BN	Batch Normalization
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
DF	Dermatofibroma
ELU	Exponential Linear Unit
FCL	Fully Connected Layer
FCLs	Fully Connected Layers
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ISIC	International Skin Imaging Collaboration
LeakyReLU	Leaky Rectified Linear Unit
MEL	Melanoma
NV	Nevus
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
Tanh	Hyperbolic tangent

VASC Vascular

1

Introduction

Contents

1.1 Motivation

According to the World Health Organization, the incidence of both non-melanoma and melanoma skin cancers has been increasing over the past decades. Currently, between 2 and 3 million non-melanoma and 132,000 melanoma skin cancers occur globally each year [15]. One in every three cancers diagnosed is skin cancer and, according to Skin Cancer Foundation Statistics, one in every five Americans will develop skin cancer in their lifetime [16]. If diagnosed on an early stage, the 5 year survival rate of the most deadly form, melanoma, can be up to 99 % [1]. However, delayed diagnosis causes the survival rate to dramatically decrease 23 % [1].

Early detection of melanoma in dermoscopy images is very important because it increases significantly the survival rate. However, the accurate recognition of melanoma is challenging due to the following reasons: low contrast between lesions and skin, visual similarity between melanoma and non-melanoma lesions, variability of shapes, colors and texture [17]. This problem has raised the attention of researchers, conducting to the development of methods for the automatic diagnosis of dermoscopy images, through techniques of pattern recognition and data classification.

The recent emergence of deep learning methods for medical image analysis has enabled the development of medical imaging-based diagnosis systems that can assist the human expert, increasing the accuracy and efficiency of their diagnoses. This development became possible due to the release of public dermoscopy datasets such as the ISIC challenges [1]. Moreover, other challenges for general images classification, such as ILSVRC [18] have contributed to the study and development of increasingly capable Convolutional Neural Networks (CNNs), which are commonly used in image classification.

There are several works based in deep learning to diagnose skin lesions' images, which achieve good results (for instance, in International Skin Imaging Collaboration (ISIC) challenges [1]). However, these works do not explore all the knowledge of skin lesions, namely their structure. It is important to explore the skin lesions' structure, because most recent skin lesions' datasets comprise different kinds of melanocytic and non-melanocytic lesions.

1.2 Hierarchical Structure of Skin Lesions

Skin lesions are organized hierarchically, where a distinction is made first between melanocytic and non-melanocytic lesion and then, between malignant and benign. At the last level of this hierarchy comes the final class of the skin lesion, as shown in figure 1.1.

Dermatologists usually divide the lesion diagnosis task into a hierarchical method: first they diagnose it as melanocytic or non-melanocytic and only then, they perform the final diagnosis (*e.g.* melanoma) [19]. The fact that dermatologists diagnose a skin lesion with a hierarchical method have inspired us to investigate whether there is any benefit to use hierarchical neural networks for lesion classification.

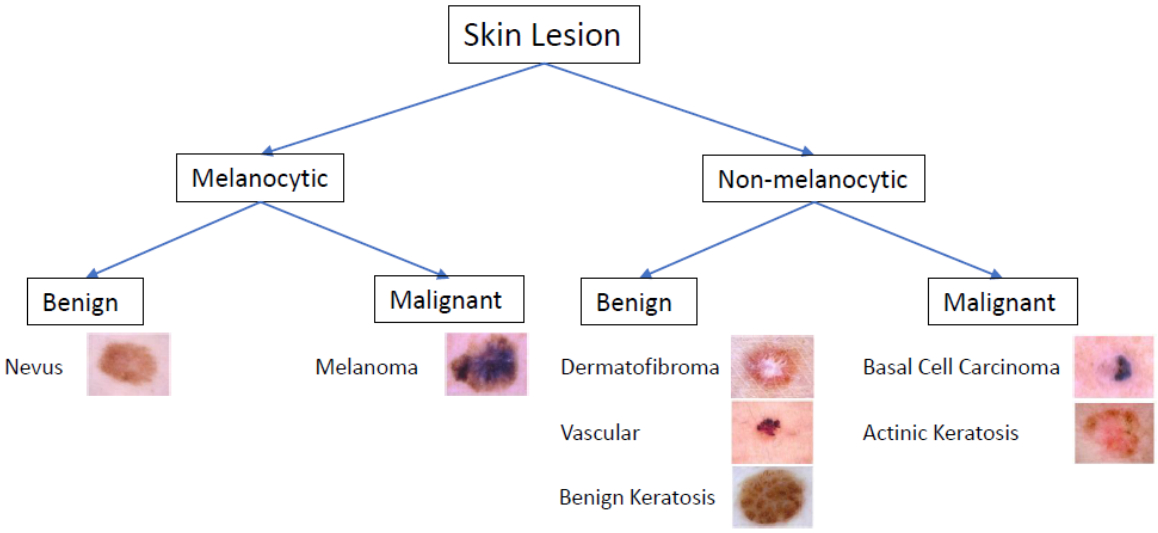
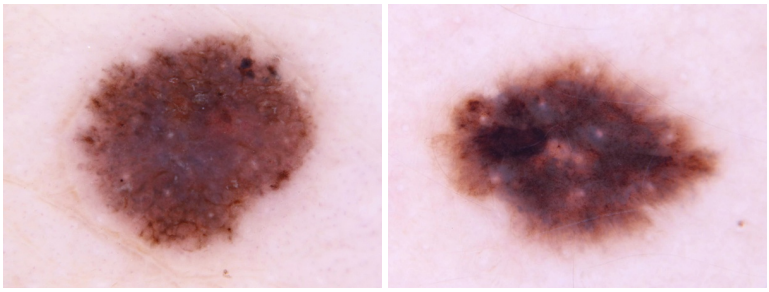


Figure 1.1: Hierarchical organization of skin lesions in ISIC 2018 [1] dataset. Dermoscopy images taken from [2].

Dermoscopy, which is a non-invasive in vivo method for the microscopic examination of pigmented skin lesions [20], is used by dermatologists to diagnose skin lesions. This technique consists in placing a fluid on the surface of the lesion, making the skin more transparent [21], and using a dermatoscope to inspect the lesion magnified. Dermoscopy has been shown to increase diagnostic accuracy over clinical naked eye visual inspection [20].

However, the diagnostic based on a visual inspection of the lesion can be difficult even for experienced physicians, because different categories of skin lesions may be very similar. Figure 1.2 presents two images of Melanoma (MEL) and Nevus (NV) lesions, that are very similar in terms of color, and shape. These difficulties lead to the development of other methods for skin lesions' diagnosis, such as techniques of pattern recognition and data classification.



(a) MEL lesion. **(b)** NV lesion.

Figure 1.2: Example of skin lesions that are visually similar, but belonging to different categories.

1.3 Previous Works in Skin Lesions Diagnosis

In the previous years, a great effort has been made by the research community in the development of computer-aided diagnosis tools that can be used by dermatologists to classify a skin lesion [22]. Usually, these systems follow a pipeline: (i) image pre-processing to deal with images that do not have sufficient quality, (ii) lesion segmentation, (iii) feature extraction to obtain a discriminative representation of the skin lesions, (iv) feature selection to reduce the dimensionality of the feature space (optional), and (v) classification, where several classifiers have been used for the diagnosis task: decision trees, Bayesian classifiers, artificial neural networks, support vector machines (SVMs), and ensemble methods [17, 23–30]. Feature extraction is a relevant step in this process, since finding appropriate descriptors is of major importance in diagnosis. There are four feature extraction methodologies: hand-crafted (asymmetry, border, color, texture), dictionary based, deep learning, and clinically inspired [22]. Deep learning and convolutional neural networks have become the technique of choice in many computer vision problems [22], namely in dermoscopy image analysis. An indicator of this popularity is the ISIC challenge 2017 [13], where 22 out of 23 works used at least one type of CNN architecture. Table 1.1 shows different related works tested in various dermoscopy datasets and compares their features extractors, classifiers and performance.

Table 1.1: Performance of related works evaluated on PH^2, ISIC 2017 [13], and ISIC 2018 datasets [1]. The ISIC 2017 submissions are ranked according to the AUC, and the ISIC 2018 submissions according to the SE.

Dataset	Ref.	Features	Classifiers	AUC	SE	SP	ACC
PH^2	[23]	Hand Crafted + Dictionary	SVM	-	98.0%	90.0%	-
	[24]	Dictionary	Random Forests	-	100%	93.0%	-
	[25]	Hand Crafted	kNN	-	96.0%	83.0%	-
ISIC 2017	[26]	Deep Learning (ResNet-50)	Softmax	91.1%	85.6%	81.2%	81.6%
	[27]	Deep Learning (ResNet-50)	Softmax	91.0%	14.0%	99.8%	84.9%
	[28]	Deep Learning (Inception_v4)	SVM	90.8%	45.1%	97.0%	88.3%
ISIC 2018	[29]	Deep Learning (ensemble)	Softmax	98.3%	83.3%	98.6%	95.8%
	[30]	Deep Learning (ensemble)	Random Forests	98.7%	80.9%	98.4%	97.2%
	[17]	Deep Learning (DenseNet)	Softmax	98.0%	78.9%	97.6%	96.9%

These datasets have different degrees of difficulty which influences the results. For ISIC 2017, we selected the three top submissions in the leaderboard ranking for melanoma and seborrheic keratosis classification combined, while for ISIC 2018 we selected three works in the top-7 of the leaderboard. Table 1.1 shows a trend: there is a convergence towards the use of deep learning features. From the results of ISIC 2017 and ISIC 2018 challenges, it is possible to postulate that deep learning features seems to be the best for the melanoma problem [13]. Unlike the traditional methods for feature extraction, implementing a feature extractor using deep learning does not require careful engineering and considerable domain expertise, because the features are learned from data using a general-purpose

learning procedure [31]. Therefore, this is rapidly becoming the most common used method for feature extraction, in skin lesions' images classification.

A previous work [32] have explored the hierarchical organization of skin lesions, in order to develop a deep learning system that performs a structured classification. The results obtained for ISIC 2017 dataset in this work showed that structured classification based on a distinction between malignant and benign lesions, followed by the diagnosis of the latter in different classes, leads to better results. However, these results remain to be validated on a larger dataset, comprising more classes of non-melanocytic lesions.

1.4 Objectives

The main goal of this thesis is to compare the performance of flat classifiers, that try to classify the lesions directly into one of the many possible categories, and hierarchical classifiers, where the classification involves more than one decision, regarding skin lesions classification. These classifiers will be implemented with deep learning methods, which have been shown to be the best methods in skin lesions classification, at ISIC 2017 and ISIC 2018. The dermoscopy images and the corresponding ground truth used in this work were extracted from the ISIC 2018 challenge website [33]. It is very important to improve the efficiency with respect to the available data, and this work also aims to study the effects of reducing the amount of data on the models' performance.

1.5 Thesis Outline

This document is organized as follows: In chapter 2 the background regarding deep learning is introduced. It starts with an introduction to machine learning, and the relevance of deep learning currently. Afterwards, a detailed explanation of CNNs is provided, and the chapter ends with the review of popular CNNs. Chapter 3 presents a brief description of the main stages regarding this thesis. Afterwards, a detailed description of the implemented models is provided. Chapter 4 starts with a description of the dataset, metrics and computational environment used in this work. Then, it presents the experimental results and discussion regarding dermoscopy images classification. Finally, in chapter 5, conclusions and future work topics are presented.

2

Deep Learning Background

Contents

In this chapter, a background of deep learning is introduced. First, an introduction to machine learning is provided. Afterwards, a detailed explanation about CNNs and a review of popular Convolutional Neural Network (CNN) architectures is presented.

2.1 Machine Learning

Machine Learning provides computer algorithms that are able to learn from data through experience [34]. According to [35], learning has the following definition:

> "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*".

A major part of the algorithms can be broadly categorized as unsupervised or supervised by what kind of information they are allowed to have during the learning process. There are also reinforcement learning algorithms that do not fit in these categories, but they are not used in this work.

Supervised learning requires a dataset containing the input variables, and the corresponding output labels [34], and the algorithm is used to learn the relation between them.

Unsupervised learning algorithms experience a dataset containing many features, but lacking labels (the output variables) [34]. In unsupervised learning, the algorithm explores the data and try to infer patterns.

Our work belongs to the supervised learning domain.

2.2 Relevance of Deep Learning

In the past decades, constructing a pattern-recognition or machine learning system required careful engineering and considerable domain expertise to design a feature extractor, which is responsible to transform the raw data into a suitable internal representation [31].

Deep learning is a part of a broader family of machine learning algorithms. A deep learning architecture consists in a stack of simple modules, which are used to progressively extract higher level features from the raw input.

The importance of this technique is that the features do not need to be designed by humans with considerable domain expertise, because they are learned from data using a task-driven learning procedure [31]. This technique takes advantage of increases in the amount of available computation and data and it is becoming state-of-the-art in many computer vision tasks.

Throughout the years, several deep learning based algorithms and architectures have been developed: recurrent neural networks, CNNs, autoencoders, reinforcement learning neural networks, etc [36]. Due to the importance of CNNs in this work, the section 2.4 provides a detailed explanation about CNNs.

2.3 Artificial Neural Networks and Multi-Layer Perceptrons

Artificial Neural Networks (ANNs) are computing systems vaguely inspired by the biological neural networks that constitute animal brains [37]. An Artificial Neural Network (ANN) is based on a collection of connected units called artificial neurons, which loosely model the neurons in a biological brain [34]. Figure 2.1 illustrates a biological neuron and the corresponding mathematical model of a neuron.

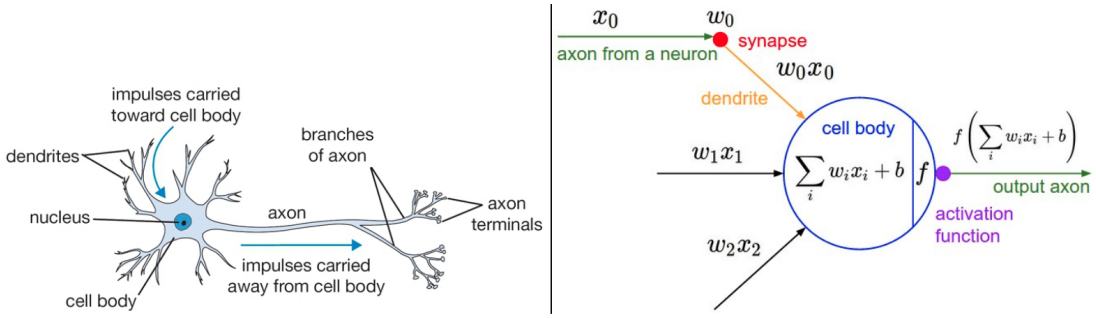


Figure 2.1: Example of biological neuron and the corresponding mathematical model. Each neuron receives weighted input signals from its dendrites and produces output signals along its axon. The axon eventually branches out and connects via synapses to dendrites of other neurons. Image retrieved from [3].

This model has a linear part followed by a non-linearity and is mathematically represented by

$$y = f\Big(\sum_i w_i x_i + b\Big), \tag{2.1}$$

where $f(.)$ denotes the non-linear activation function, w_i represents a weight, x_i denotes the input to the neuron, and b is the bias term. An ANN is obtained by connecting artificial neurons. Figure 2.2 illustrates a multi-layer perceptron, which is an example of ANNs. The multi-layer perceptron's neurons are structured in layers, and every pair of neurons belonging to consecutive layers are connected.

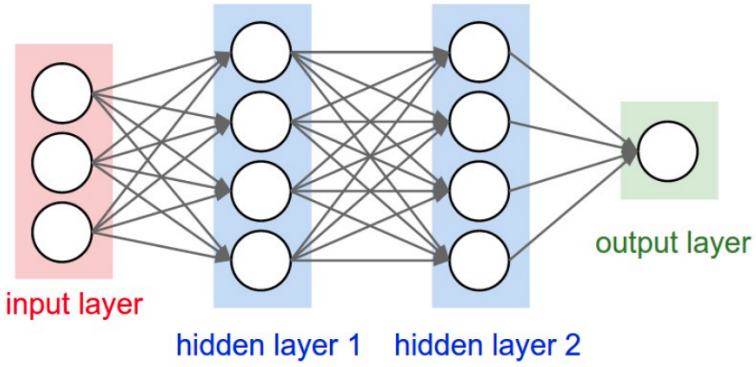


Figure 2.2: Example of a multi-layer perceptron that comprises many artificial neurons, represented with the circles. Image retrieved from [3].

The neural network receives as input an array, which will be processed along the hidden layers, by propagating the information sequentially from the input layer to the last layer, which is called the output layer [38].

2.4 Convolutional Neural Networks

ANNs are used to process vectorized inputs. CNNs are a specialized kind of neural network for processing data that has a known grid-like topology, like images [34]. Its name indicates that the network employs a mathematical operation called convolution.

In this section, a formal definition of convolution applied to images will be provided. Afterwards, the most common kinds of layers in CNNs will be explained.

2.4.1 Convolutional Layer

An image I is a set of pixels in a 3D-array format, with a width $W(I)$, height $H(I)$ and depth $D(I)$. The convolution operation, to the image I, is performed by a kernel K with a width $W(K)$, height $H(K)$ and depth $D(K) = D(I)$. Mathematically, this operation is formally defined by the following expression [34]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{2.2}$$

In this expression, the depth variable is omitted, since the filters slide spatially across the image, and not across its channels.

The input of a convolutional layer is an image I. Initially, multiple convolutions are performed between the image I and different kernels. The outputs of each convolution are then concatenated, forming a

new image I'. Afterwards, a non-linear activation function is applied to every pixel in the new image I', resulting in the output of the convolutional layer, often called feature map. In the following section the most common activation functions are presented.

2.4.2 Non-linear Activation Functions

The selection of non-linear activation functions is an important decision, regarding the training of any type of neural network, because it can speed up or stagnate network learning, depending on whether the function is good or bad, respectively.

Table 2.1 describes the most common activation functions used in deep learning: Hyperbolic tangent (Tanh), Sigmoid, Rectified Linear Unit (ReLU), Leaky Rectified Linear Unit (LeakyReLU), and Exponential Linear Unit (ELU) [14].

Sigmoid and Tanh have saturated zones with small gradients, and require the calculation of the exponential, which is computationally expensive. ReLU is a faster learning activation function that offers the best performance and generalization in deep learning when compared with Sigmoid and Tanh, and has been the most widely used activation function for deep learning applications with state-of-the-art results to date [14]. However, it creates dead neurons (for $z < 0$), which may lead to cases where the neuron will never reactivate and further, like vanishing gradients problem, the learning starts to slow [39]. LeakyReLU introduces a small negative slope to the ReLU, to sustain and keep the weight updates alive during the entire propagation process, however there is not a significant improvement in the performance, when compared to ReLU [14]. ELU can alleviate the vanishing gradient problem and it improves the learning characteristics, guaranteeing state-of-the-art results compared to ReLU [14].

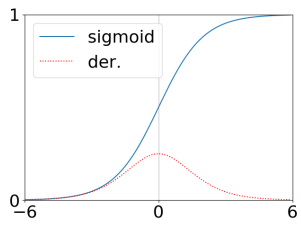
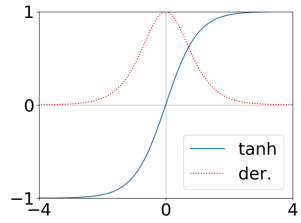
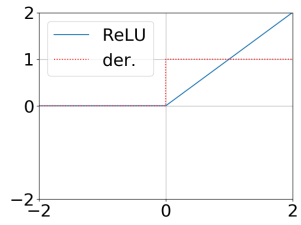
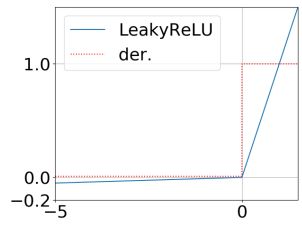
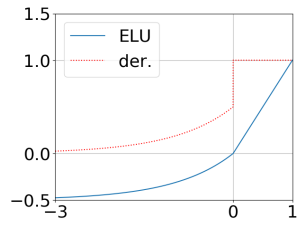
2.4.3 Pooling Layer

A pooling layer receives as its input an image I with a width $W(I)$, height $H(I)$ and depth $D(I)$, and it is characterized by a window with width $W(Win)$ and height $H(Win)$. This window will slide across the image, with a specific stride, and for each $H(Win) \times W(Win) \times D(I)$ block of pixels selected, a pooling function is applied. Each pooling function replaces the input image by a summary statistic of the nearby inputs [34]. For instance, the maximum pooling operation reports the maximum input within a rectangular neighborhood, defined by the window.

Other popular pooling functions include the average pooling of a rectangular neighborhood, the L^2 norm of a rectangular neighborhood, or a weighted average based on the distance from the central pixel [34]. In this work, the maximum and average pooling functions are the most used.

This operation will transform the input image, I, into another image, I', with reduced spatial size. If the window size increases, the output image's size decreases, improving the computational efficiency

Table 2.1: Description of the most commonly used non-linear activation functions in deep learning. Plot of the activation function and its derivative (der.). Advantages and disadvantages retrieved from [14].

Activation	Plot	Advantages	Disadvantages
Sigmoid(z) = $\frac{1}{1+e^{-x}}$		• Easy to understand.	• Gradient saturation; • Slow convergence; • Non-zero centered output.
Tanh(z) = $\frac{e^z-e^{-z}}{e^z+e^{-z}}$		• Zero centered output.	• Gradient saturation; • Produces dead neurons.
ReLU(z) = max$(0, z)$		• Faster learning; • Faster computation;	• Non-zero centered output. • Easier to overfit; • Produces dead neurons.
LeakyReLU(z) = max$(\alpha z, z)$		• Only produces non-zero gradients.	• Easier to overfit; • Non-zero centered output.
ELU(z) = $\begin{cases}\alpha(e^z-1), & \text{if } z<0\\ z, & \text{if } z>=0\end{cases}$		• Pushes mean activation towards zero. • Faster learning and better generalisation compared to ReLU and LeakyReLU.	• Non zero-centered output;

of the network because the next layer processes an input with smaller dimension. It also helps to make the representation approximately invariant to small translations of the input [34].

2.4.4 Fully Connected Layer

The last layers of a CNN usually comprise Fully Connected Layers (FCLs). Before the first Fully Connected Layer (FCL) of a CNN, there is an operation that flattens the output of the previous layer (converts the 3D-array into a single 1D-array).

Each FCL connects each neuron to every neuron in the previous layer. It converts one input 1D-array X into one output 1D-array Y. Each element (neuron) in Y is obtained by a linear combination of all elements in X. Finally, a non-linear activation function is applied to all elements of Y, resulting in the layer output. Thus, the ANNs described previously are neural networks exclusively comprised by FCLs.

The last layer of a neural network for classification is a FCL with a softmax as non-linear activation function. The number of neurons in this layer is the number of different classes for the given problem. The softmax function $\sigma : \mathbb{R}^{\mathrm{C}} \to \mathbb{R}^{\mathrm{C}}$ is defined by the formula:

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}, \tag{2.3}$$

for $i = 1, ..., C$ and $\vec{x} = (x_1, ..., x_C) \in \mathbb{R}^{\mathrm{C}}$, where C is the number of classes. Each $\sigma(\vec{x})_i$ represents the probability that the input image belongs to the i-th class computed by the CNN. Thus, we have that

$$\sum_{i=1}^{C} \sigma(\vec{x})_i = 1. \tag{2.4}$$

Most of the works of classification problems have adopted the cross entropy loss [40], which is given by

$$L(y, \hat{y}) = -\sum_{c=1}^{C} y_{o,c} \log \hat{y}_{o,c} \qquad [41], \tag{2.5}$$

where C is the number of classes, $y_{o,c}$ is a binary indicator that takes the value of 1 if the class label c is the correct for observation o, and 0 otherwise. $\hat{y}_{o,c}$ is the predicted probability that observation o is of class c.

2.4.5 Optimization

The optimizer defines the update rules for the parameters in order to minimize the loss function. The optimization is performed with the gradient algorithm:

$$w_{i+1} = w_i + \Delta w_i, \qquad \Delta w_i = -\eta \nabla_w f(w)|_{w_i}, \tag{2.6}$$

where η is the learning rate and the gradient $\nabla_w f(w)|_{w_i}$ is typically calculated, using the backpropagation algorithm [34]. This algorithm forwards the input through the network and computes the output. Then, the loss value is computed and the algorithm applies the chain rule, from the output to the input layer, in order to compute each parameter's contribution in the loss [42]. Thus, Δw_i is obtained and used to update the weights for the next iteration.

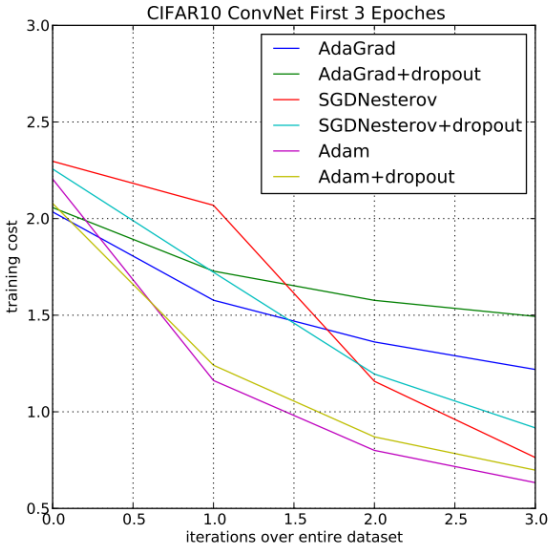
An epoch is achieved when all the training samples have been used once to update the parameters. There are three training modes [34], that defines how the gradient descent method is used:

1. Batch: This training mode is related with the batch gradient descent, where the parameters are updated with the average of the gradients of all the training examples. When the dataset is huge, this method is not efficient.

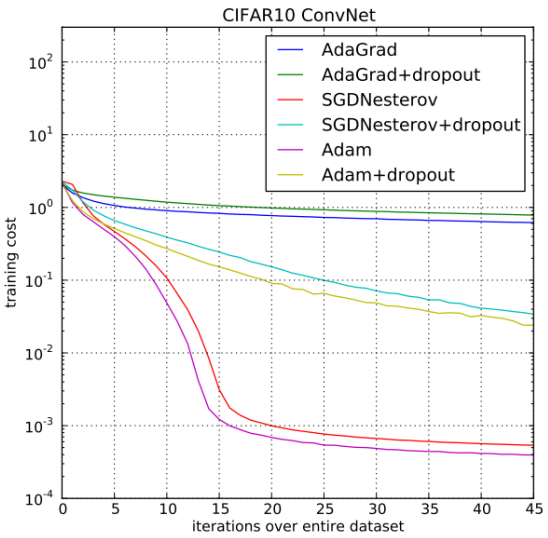
2. Online: This training mode is associated with the stochastic gradient descent, where the parameters are updated with a single example in the training set, randomly selected.

3. Mini-batch: This training mode is related with the the mini-batch gradient descent, where a batch of a fixed number of training examples, which is less than the dataset's size, is used. This variant achieves the advantages of the previous ones, and is the most commonly used to train a neural network [34].

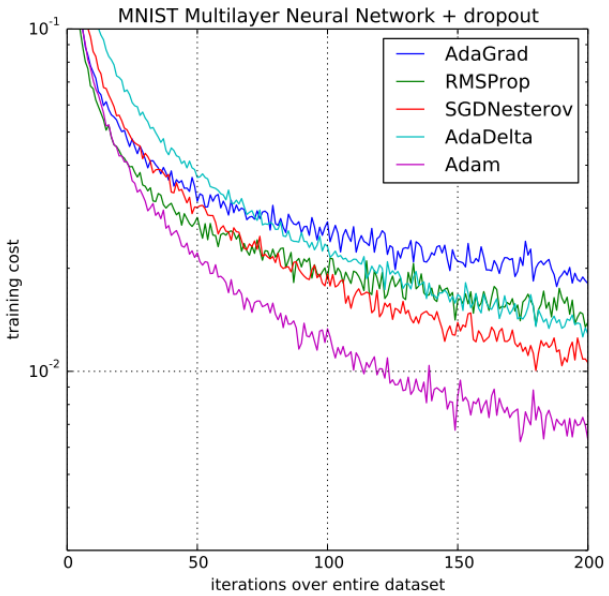
The most commonly used gradient descent variants are Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), RMSProp, AdaDelta, and AdaGrad [34]. The paper [6], which introduces the Adam, compares these optimizers and shows that Adam outperforms the others on the classification of MNIST [4] and CIFAR10 [5] datasets, since with the same number of iterations it achieves a smaller training cost, as shown in figure 2.3.



(a) Comparison of optimizers at CIFAR10, after 3 epochs of training.



(b) Comparison of optimizers at CIFAR10, after 45 epochs of training.



(c) Comparison of optimizers at MNIST, after 200 epochs of training.

Figure 2.3: Comparison of AdaGrad, RMSProp, SGD with Nesterov momentum, AdaDelta, and Adam at MNIST [4] and CIFAR10 [5] datasets. Images retrieved from [6]

Algorithm 2.1: Adam algorithm for stochastic optimization. g_t^2 indicates the element-wise square $g_t \cdot g_t$. All operations on vectors are element-wise. β_1^t and β_2^t denotes β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1]$: Exponential decay rates for the moment estimates
Require: $f(w)$: Stochastic objective function with parameters w
Require: w_0: Initial parameter vector
$m_0 = 0$ (Initialize first moment vector)
$v_0 = 0$ (Initialize second moment vector)
$t = 0$ (Initialize timestep)
while w_t not converged **do**
  $t = t + 1$
  $g_t = \nabla_w f_t(w_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
  $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_t = m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\hat{v}_t = v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $w_t = w_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end
return w_t (Resulting parameters)

The algorithm 2.1 , retrieved from [6], presents the Adam algorithm for stochastic optimization. The Adam algorithm, for gradient-based optimization of stochastic objective functions, is computationally efficient and combines the advantages of two popular optimization methods: the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objective functions. [6].

2.5 State-of-the-Art CNNs

In this section, a review of the most popular CNN architectures, designed for image classification, is provided.

2.5.1 AlexNet

AlexNet [7] was published by Alex Krizhevsky in 2012. This paper is considered one of the most influential papers published in computer vision. As of 2020, it has been cited over 58,000 times. The figure 2.4 is an illustration of AlexNet architecture.

The neural network, with 60 million parameters to be estimated, receives as input an image 224 $\times$ 224 $\times$ 3 and it is composed by 5 convolutional layers and 3 fully connected layers (including the softmax classifier with 1000 neurons). ReLU is the non-linear activation function used in every other hidden layer.

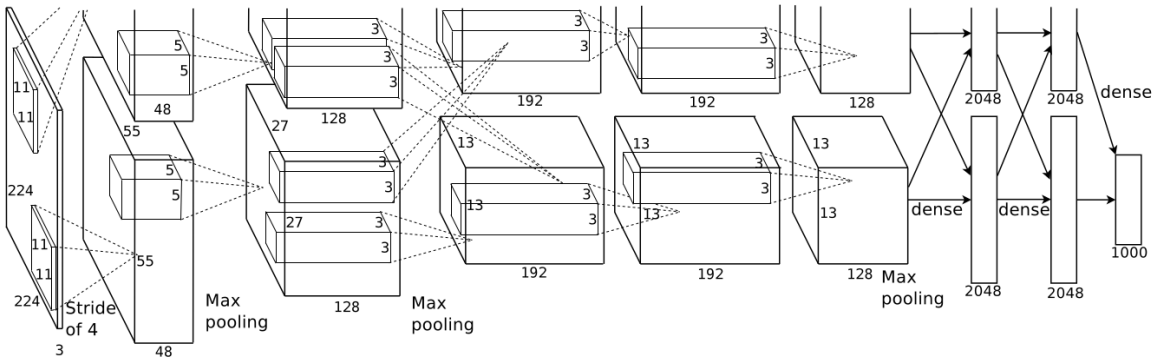


Figure 2.4: An illustration of the architecture of AlexNet CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. Image retrieved from [7]

2.5.2 GoogLeNet

GoogLeNet [8] was published in 2015 and proposes a deep CNN codenamed Inception. This network was carefully crafted designed in order to increase the depth and width, without increasing the computation budget, so that it could be used practically in the real world. The inception module is presented in figure 2.5.

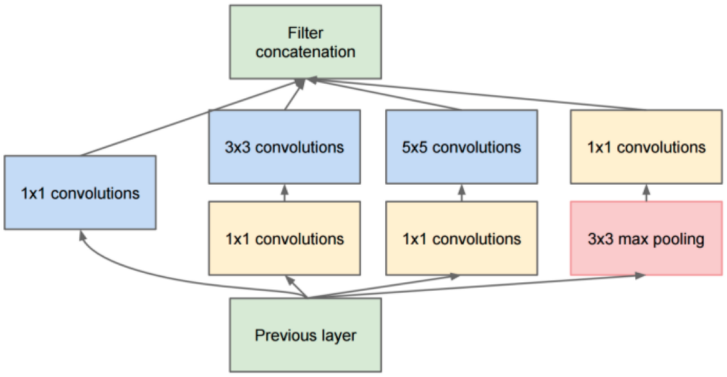


Figure 2.5: Inception module. Image retrieved from [8]

Small filters help to capture local details and features, whereas spread out features of higher abstraction are captured with larger filters. Instead of choosing one size for the filters in each layer, multiple parallel convolutions are performed with different size filters as well as maximum pooling, and the resulting feature maps are concatenated into one big feature map that is fed as input to the next inception module.

Before the 3×3 and 5×5 convolutions, a 1×1 convolution is performed to the input feature map[1]. This helps to reduce the depth of the feature map before applying the convolutions with larger filters,

[1] the depth is omitted since it depends on the depth of the previous layer

decreasing the number of operations. Furthermore, it adds extra non-linearity. The 1 $\times$ 1 convolutions added after the 3 $\times$ 3 maximum pooling helps to control the depth of the output feature map. By stacking these modules and maximum pooling layers, the authors designed a deep neural network called GoogLeNet with 22 layers. The GoogLeNet architecture is illustrated in figure 2.6

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 2.6: GoogLeNet architecture. Image retrieved from [8].

2.5.3 VGG

The paper [9] written by Karen Simonyan and Andrew Zisserman (2015) investigates the effect of the CNN's depth on its accuracy in the large-scale image recognition setting. This paper presents six CNNs configurations and compares their accuracies when applied to image classification. Figure 2.7 presents these configurations.

During the training stage, 224 $\times$ 224 $\times$ 3 input images were used. To obtain them, they were randomly cropped from rescaled training images. Also, to further augment the training set, the crops underwent random horizontal flipping and random RGB colour shift. Training images rescaling was performed with and without scale jittering.

At test time, test set augmentation was done by horizontal flipping the images and the softmax class posteriors of the original and flipped images are averaged to obtain the final scores for the image.

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

Figure 2.7: Convolutional networks configurations. The depth of the configurations increases from the left (A) to the right (E). The bold layers represent the added layers to the previous configuration. The convolutional layer parameters are denoted as "conv ⟨ receptive field size ⟩ - ⟨ number of channels ⟩". Image retrieved from [9].

2.5.4 ResNet

The paper [10] presents a residual learning framework to ease the training of networks that are substantially deeper than those presented before. This approach addresses the performance degradation problem: with the network depth increasing, the accuracy gets saturated and then degrades rapidly. This problem is not caused by overfitting, since the training error also increases by adding more layers.

The innovation in this paper was the introduction of a new building block shown in figure 2.8, to design CNNs.

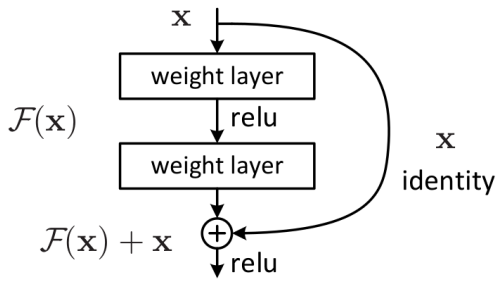


Figure 2.8: Residual learning: a building block. Image retrieved from [10]

Unlike the previous networks, instead of stacking a few layers directly to fit a desired mapping $H(x)$,

the authors explicitly let these layers fit a residual mapping $F(x) = H(x) - x$, which they hypothesized to be easier to optimize than $H(x)$. This can be realized by a feedforward neural network with "shortcut connections" (connections skipping one or more layers). $F(x)$ is defined as the residual mapping to be learned.

The building block shown in figure 2.8 is formally defined as:

$$y = F(x, W_i) + x. \tag{2.7}$$

Here x and y are the input and output vectors of the layers considered. The function $F(x, W_i)$ represents the residual mapping to be learned and its form is flexible. The dimensions of x and F must be the same in (2.7). If this is not the case, a linear projection W_s is performed by the shortcut connections to match the dimensions:

$$y = F(x, W_i) + W_s \cdot x. \tag{2.8}$$

The authors of this work performed some experiments to compare plain and residual networks results which will be described below.

The plain baselines were mostly inspired by the philosophy of VGG networks [9]. The residual networks were based on the equivalent plain networks with shortcut connections. The identity shortcuts can be directly used when the input and the output are of the same dimensions.

Two plain networks were evaluated with 18 and 34 layers. The degradation problem was observed, since the 34-layer plain network had a larger error than the 18-layer plain network.

Then, two residual networks were evaluated with 18 and 34 layers with the same baseline architectures as the above plain networks. In this situation, the 34-layer residual network has a smaller training/validation errors than the 18-layer residual network. This indicates that the degradation problem was well addressed and shows the effectiveness of residual learning on extremely deep systems.

2.5.5 DenseNet

Paper [11] presents a neural network called DenseNet where each layer connects to every other layer in a feed-forward fashion in order to alleviate the vanishing-gradient problem and strengthen feature propagation.

The architecture proposed in this paper is an attempt to ensure maximum information flow between layers in the network. All layers are connected directly with each other as shown in figure 2.9.

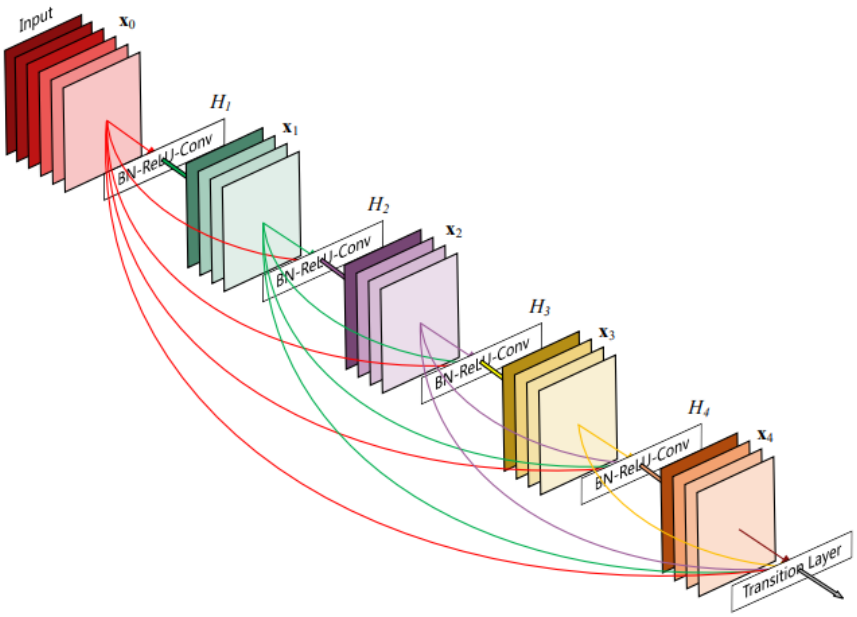


Figure 2.9: A 5-layer dense block with a growth rate of $k = 4$. Image retrieved from [11]

Besides better parameter efficiency, one big advantage of DenseNets is their improved flow of information and gradients throughout the network, which makes them easy to train. The output of the l^{th} layer in DenseNet is given by

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{2.9}$$

where $[x_0, x_1, ..., x_{l-1}]$ refers to the concatenation of the feature-maps produced in layers $0, ..., l-1$. The authors defined $H_l(.)$ as a composite of three consecutive operations: Batch Normalization (BN), followed by a ReLU and a 3×3 convolution.

To facilitate down-sampling layers, that change the size of feature-maps, they divide the network into multiple densely connected dense blocks. The layers between blocks are transition layers, which do 1×1 convolution and pooling.

On ImageNet [18], the authors used a DenseNet with bottleneck layers (1×1 convolution introduced before each 3×3 convolution) and compression (reduce the number of feature-maps at transition layers) on 224×224 images. The exact network configurations used on ImageNet are shown in figure 2.10.

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Figure 2.10: DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv. Image retrieved from [11]

2.5.6 CNNs Evolution

With the release of public and larger image datasets, and the evolution of computers' computational capabilities, the researchers started to implement deeper neural networks and to use smaller size filters (normally 3×3 - the minimum to capture spatial context), as [9] confirmed that a deep network with small filters outperforms a shallow network with larger filters. For instance, AlexNet [7] (2012) comprised 8 layers with filters' size up to 11×11, VGG-E [9] (2015) comprised 19 weight layers with filters' size up to 3×3, and DenseNet [11] (2017) had a model with 201 layers, using other techniques, with filters' size up to 3×3, with better results. Typically, increasing the size of a neural network translates into increasing the number of operations required to train the network. The evolution in the computational power of computers allows the training of larger networks in a shorter time.

In addition to a good network architecture design, there are other techniques that can optimize the results. Some of the techniques used in these papers are: dropout [7–9,11], data augmentation [7,9–11], scale jittering [9] and ensemble of models [8–10].

3

Methodology

Contents

As stated previously, the main goal of this work is to assess the benefits of using a hierarchical classifier to diagnose a skin lesion, in order to mimic a dermatologist's procedure. This chapter presents and explains the methodologies used to achieve this goal.

3.1 Main Stages

The main stages of this thesis are presented in figure 3.1.

Figure 3.1: Main stages of this thesis. The final stage of evaluating the results are omitted since it is common to every work.

The first stage is the image acquisition as well as the corresponding ground truth, which was obtained at ISIC 2018 webpage [33]. This dataset comprises 10,015 images with ground truth and 1,512 images without the ground truth data.

In the second stage, the extracted dataset was processed in order to obtain the final dataset, which was used for training and validation. First, the original dataset was randomly divided into a training and a validation sets, which represented about 80 % and 20 % of the original, respectively. Then, in order to prevent overfitting, data augmentation was performed to the training set. All images were resized to 224 $\times$ 224.

The third stage is the implementation of several CNN architectures to diagnose the skin lesions into the following seven different classes: Melanoma (MEL), Nevus (NV), Benign Keratosis (BKL), Dermatofibroma (DF), Vascular (VASC), Basal Cell Carcinoma (BCC), and Vascular (VASC). First, flat classifiers were implemented using several state-of-the-art architectures and their performance were compared. Then, hierarchical models were implemented and their performance was compared against the one obtained by the corresponding flat classifier. Finally, mixed models of a flat and a hierarchical models were implemented in order to capture the strengths of each approach. The implementation of these models will be explained in detail in this chapter.

3.2 Flat Classifier

A flat classifier performs a single decision, whose output includes all the final categories [43]. Figure 3.2 presents the flat classifier's architecture.

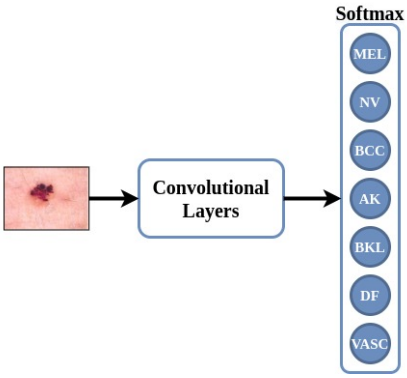


Figure 3.2: Architecture of the flat classifier. The last layer is a FCL with seven units, corresponding to the seven different categories in the dataset used in this work.

This model receives a skin lesion image as input and comprises several convolutional layers inspired in state-of-the-art architectures. The last layer is a FCL whose number of units is the same of different categories in the dataset, and each unit represents the probability that the lesion belongs to the corresponding category. To reduce the probability of the network memorizing the training set and to improve generalization, dropout with 50 % probability was applied before the decision layer as in [32].

Three flat classifiers were implemented, each one based on a state-of-the-art CNN architecture, in order to analyse and compare their performance: VGG-19 [9], ResNet-101 [10], and DenseNet-121 [11]. These CNNs were chosen due to their great results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) challenge [18].

Each of the models was trained using two approaches: from scratch and with transfer learning [44]. The models trained from scratch ran for 50 epochs and their weights were initialized with the Glorot uniform initializer [45] (also called Xavier uniform initializer). This initializer assigns each weight with a value that follows a uniform distribution within $[-a, +a]$, where a is given by:

$$a = \sqrt{\frac{6}{n_{in} + n_{out}}}, \tag{3.1}$$

where n_{in} and n_{out} are the number of input and output units in the weight tensor, respectively.

The models trained with transfer learning were initialized with the weights from the model pre-trained on the ImageNet dataset [18], and fine-tuned for 20 epochs.

For all the models, the training was carried out by optimising the categorical cross entropy, using the Adam optimizer, with a mini-batch of size 10. The number of epochs was chosen with the validation set, *i.e.*, the model loaded the weights achieved in the epoch that led to the minimum cross entropy loss in the validation set.

3.3 Hierarchical Model

The hierarchical model exploits the hierarchical organization of skin lesions, as shown in figure 1.1. When predicting the category of each skin lesion image, the model makes intermediate decisions, and then predicts the final output, mimicking the approach followed by a dermatologist. There are three levels of decision: (i) melanocytic vs. non-melanocytic, (ii) benign vs. malignant, and (iii) final category.

Figure 3.3 illustrates the hierarchical inference procedure of a skin lesion in our dataset, with five different classifiers (see figure 1.1). Classifiers **(a)** and **(c)** perform intermediate decisions and the remaining classifiers perform the final classification of skin lesions. Since our dataset has only one category in melanocytic benign lesions and only one category in melanocytic malignant lesions, the decision level benign vs. malignant can be omitted on this case.

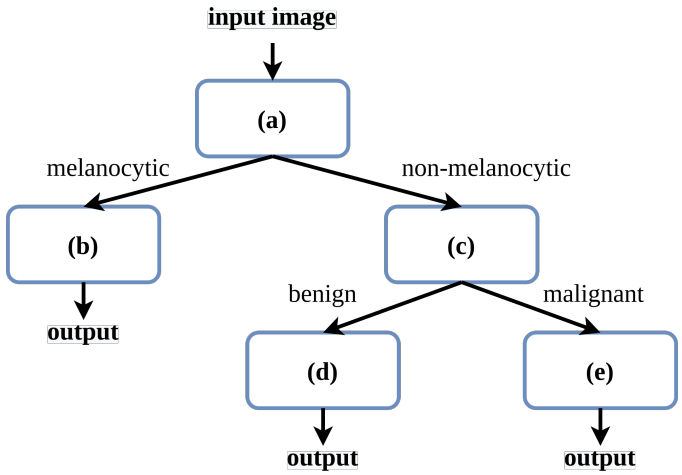
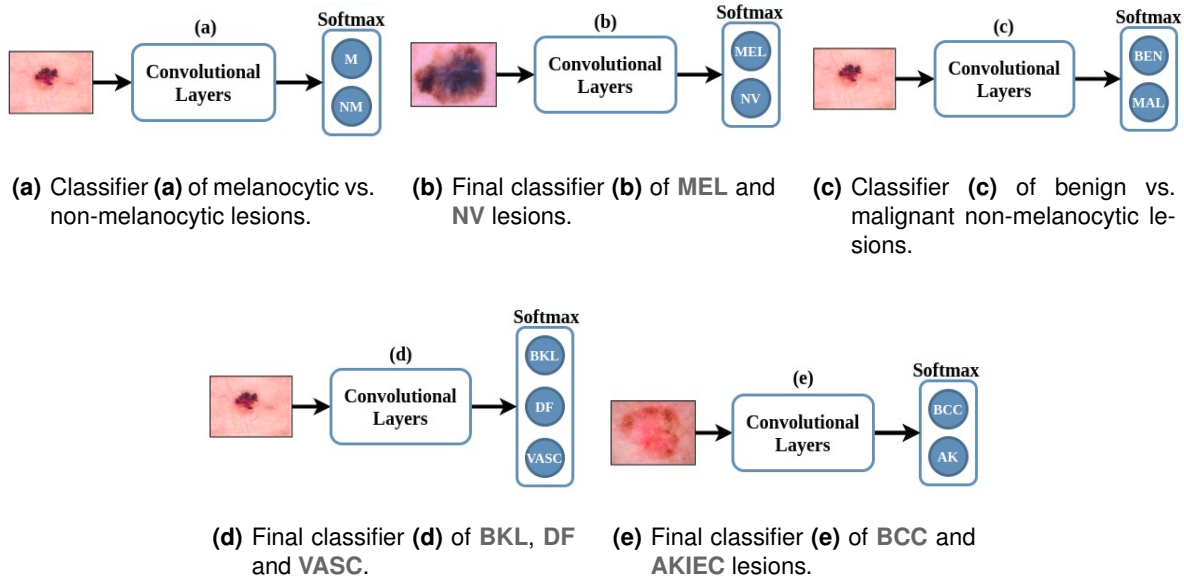


Figure 3.3: Hierarchical inference procedure of a skin lesion. For simplicity, we refer to a classifier by its letter. For instance the classifier that diagnoses non-melanocytic malignant lesions will be referred by **(e)**.

In this work, two approaches were considered with respect to the architecture of the hierarchical's five classifiers: (i) non-shared features among the five classifiers, and (ii) shared features among the five classifiers. These approaches will be explained in the detail in the following sections.

3.3.1 Non-shared Features

When the features are not shared among the five classifiers, the model comprises five different networks, each one with a CNN as feature extractor, and a FCL responsible for the decision. This architecture allows each classifier to be trained independently with the corresponding subset of the training set. For instance, the training set used at classifier **(e)** comprises all the non-melanocytic malignant lesions. Figure 3.4 illustrates the architectures of the five different networks, belonging to the hierarchical model.



(a) Classifier **(a)** of melanocytic vs. non-melanocytic lesions.

(b) Final classifier **(b)** of **MEL** and **NV** lesions.

(c) Classifier **(c)** of benign vs. malignant non-melanocytic lesions.

(d) Final classifier **(d)** of **BKL**, **DF** and **VASC**.

(e) Final classifier **(e)** of **BCC** and **AKIEC** lesions.

Figure 3.4: Architecture of each network, belonging to the hierarchical model with non-shared features among the five classifiers. The

The training procedure was done with the same strategies as the flat classifier's training (*i.e.*, optimising the categorical cross entropy with the Adam optimizer, during 20 epochs, with the same learning rate scheduler, and using the pre-trained model in ImageNet). For each network, dropout was applied with 50 % probability, before the decision layer. The training sets used to train each classifier are described in the table 3.1.

Table 3.1: Training set description of each CNN belonging to the hierarchical classifier

CNN	Training set images	No. training images
(a)	Melanocytic or non-melanocytic (all)	24,012
(b)	MEL or NV	18,633
(c)	Non-melanocytic	5,379
(d)	BKL, DF or VASC	3,273
(e)	BCC or AKIEC	2,106

This implementation trains the five networks independently, and needs to store their weights, which is expensive in terms of time and memory. The following approach makes the hierarchical classifier more efficient in terms of memory and training duration.

3.3.2 Shared Features

In this approach, a single CNN is used as a feature extractor, and the five classifiers (FCLs) are stacked on top of the feature extractor, as shown in figure 3.5. Dropout was applied with 50 % probability, before each decision layer.

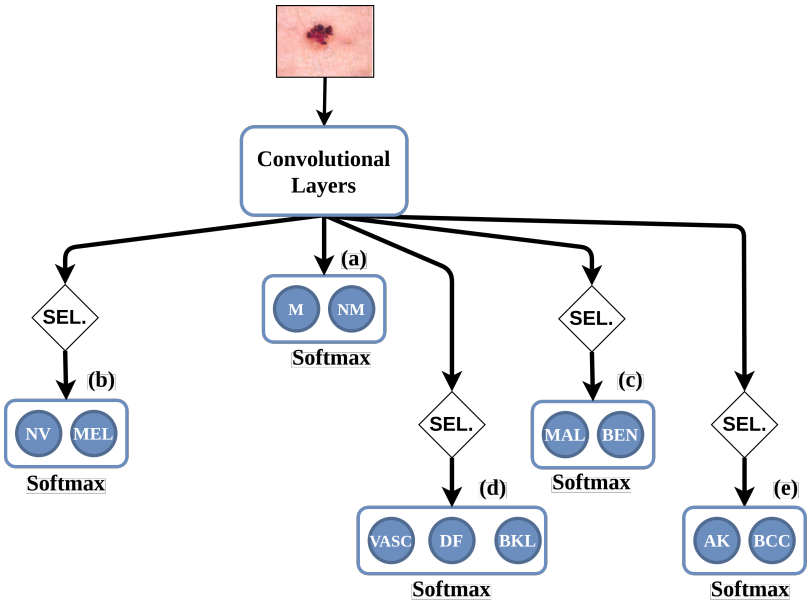


Figure 3.5: Training architecture of the hierarchical model, with a single feature extractor.

Since the features are shared by the five classifiers, there is a single model to train with the whole training set. However, this approach poses an additional challenge: each kind of lesion should only update a part of the classifiers (FCLs). For instance, the category VASC, which is a non-melanocytic and benign lesion, should only update the classifiers **(a)** (melanocytic vs. non-melanocytic), **(c)** (non-melanocytic benign vs. non-melanocytic malign), and **(d)** (final category of non-melanocytic benign lesions).

Selecting which classifiers to update, according to the lesion's category, is represented in figure 3.5 with the "SEL." blocks. For each category of lesion, we set the loss to zero at the specific classifiers that should not update the parameters. Since the loss is zero, the backpropagation algorithm does not update them.

With this approach, the model has about 7 million parameters, which is five times less than the previous approach. The training phase is also less time consuming, since at each epoch the model is trained with 24,012 images, and in the previous approach at each epoch the five classifiers are trained

with a total of 53,403 images (sum of the last column in table 3.1).

In order to assess the validity of this approach before applying it to the skin lesion images, we generated a synthetic dataset with points in the cartesian plane, following gaussian distributions. This dataset comprised four classes. Table 3.2 shows the mean values, $\vec{\mu} = [\mu_x, \mu_y]$, and the covariance matrices, Σ, of the four gaussian distributions, and figure 3.6 presents the generated training data in the cartesian plane.

Table 3.2: Parameters of gaussians for each class.

Class	μ	Σ
1 - cyan	$[-6.93; 4.97]$	$\begin{bmatrix} 3.00 & 3.00 \\ 4.00 & 4.00 \end{bmatrix}$
2 - blue	$[4.09; 4.06]$	$\begin{bmatrix} 4.00 & 4.00 \\ 3.47 & 3.65 \end{bmatrix}$
3 - green	$[5.42; -5.55]$	$\begin{bmatrix} 3.45 & 3.60 \\ 3.58 & 3.55 \end{bmatrix}$
4 - black	$[-5.44; -4.75]$	$\begin{bmatrix} 3.06 & 3.16 \\ 3.96 & 4.28 \end{bmatrix}$

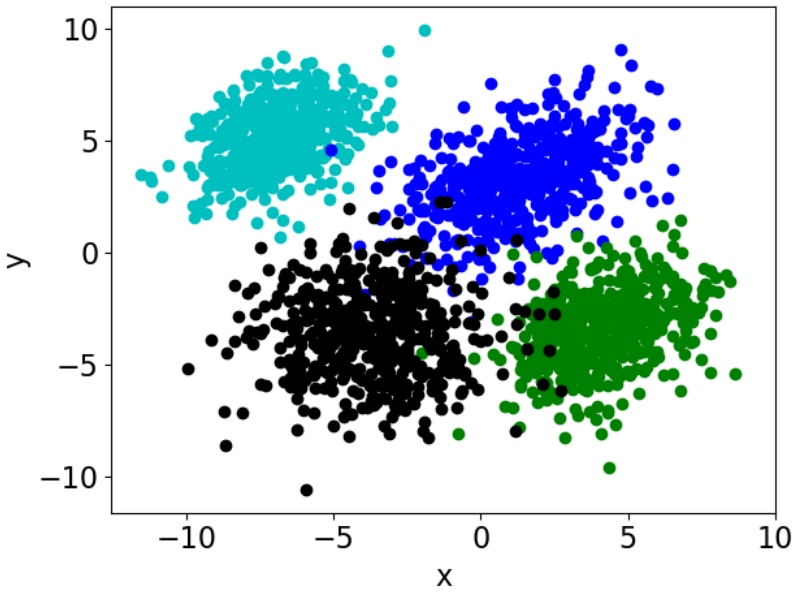


Figure 3.6: Generated training set (2000 training examples) with four gaussian distributions. Class 1 is represented in cyan, class 2 in blue, class 3 in green, and class 4 in black.

The model comprises three classifiers (FCLs) that are stacked on top of a single feature extractor: classifiers **(a)**, **(b)**, and **(c)**. The classifier **(a)** aims to split the classes 1 and 2 from the classes 3 and 4 (top classes vs. down classes). Classifier **(b)** aims to split the data that classifier **(a)** predicted to be at top (top left vs. top right), and classifier **(c)** aims to split the data that classifier **(a)** predicted to be at down (down left vs. down right). Figure 3.7 illustrates the inference stage with the three classifiers and two decision levels: (i) top vs. down, and (ii) left vs. right.

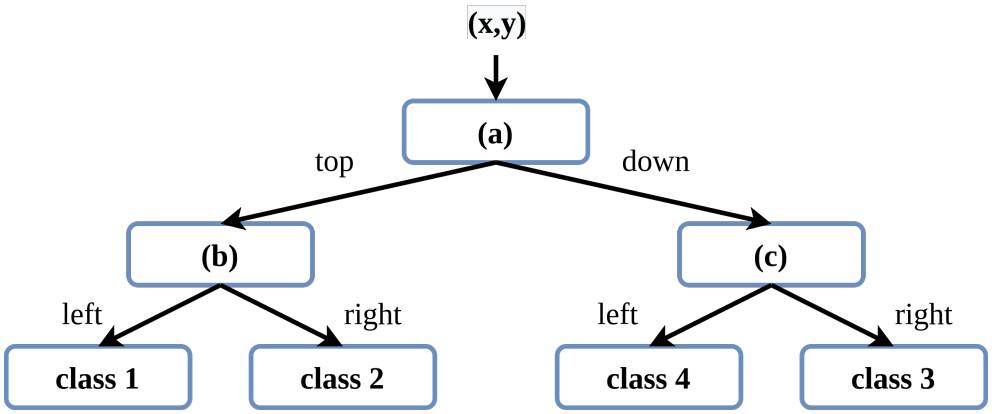
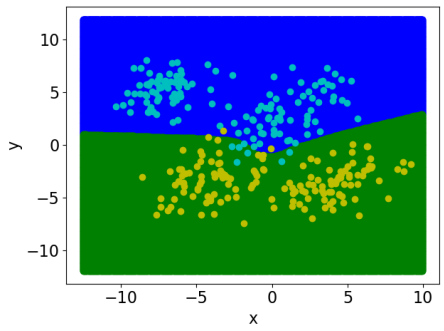


Figure 3.7: Hierarchical representation of data and the inference stage.

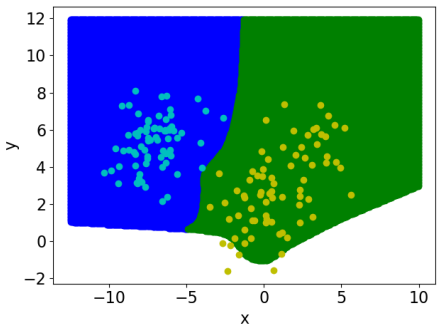
At training stage, since the features are shared by the three classifiers, classes 1 and 2 only update the weights of classifiers **(a)** and **(b)**, and classes 3 and 4 only update the weights of **(a)** and **(c)**. The feature extractor is a multi-layer perceptron with three hidden layers. The training was carried out by optimising the cross entropy loss, with the Adam optimizer, for 1,000 epochs.

After training the model, it predicted the class of every point in the cartesian plane with $x \in [-12, 10] \cup y \in [-12, 12]$. The corresponding regions predicted by each classifier are presented in figure 3.8. The validation data associated with the ground truth is also presented to verify if the predicted decision regions make sense.

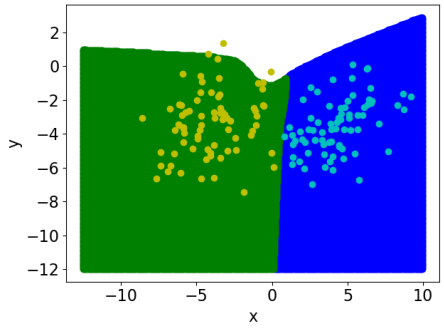
These results suggest that the hierarchical model with shared features is able to correctly learn the hierarchical structure of the synthetic dataset. It is expected that a similar approach may also learn correctly the hierarchical structure of the skin lesions.



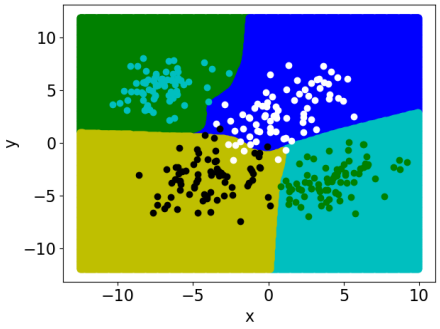
(a) Decision regions predicted by classifier **(a)**. Validation data: cyan - classes 1 and 2. Yellow - classes 3 and 4.



(b) Decision regions predicted by classifier **(b)**. Validation data: cyan - class 1. Yellow - class 2.



(c) Decision regions predicted by classifier **(c)**. Validation data: cyan - class 3. Yellow - class 4.



(d) Concatenation of the decision regions predicted by classifiers **(b)** and **(c)**. Validation data: cyan - class 1. Yellow - class 2. Green - class 3. Black - class 4.

Figure 3.8: Predicted decision regions (background colors) with the validation data (dot points). The dot points are the validation data and their colors represent their labels.

3.4 Mixed Classifier

Regarding the hierarchical models explained in section 3.3 to diagnose skin lesions, during the inference stage, for each image, they have to make more than one decision to reach the final category, and if at least one decision is incorrect, the final diagnose will be automatically incorrect. Therefore, it is very important that in each decision, the hierarchical model is very confident, to prevent error propagation. One way of evaluating the confidence of a classifier is to assess the softmax probabilities. If the classifier has a high confidence, the difference between the two largest softmax probabilities is expected to be large.

Figure 3.9 confirms this behavior, showing the relation between the proportion of correct decisions at classifier **(a)**, belonging to the hierarchical model with non-shared features, which predicts if the lesion is melanocytic or non-melanocytic, in the validation set, and the difference between the two softmax probabilities of its output layer. The proportion of correct decisions is relative to the number of decisions made in each interval.

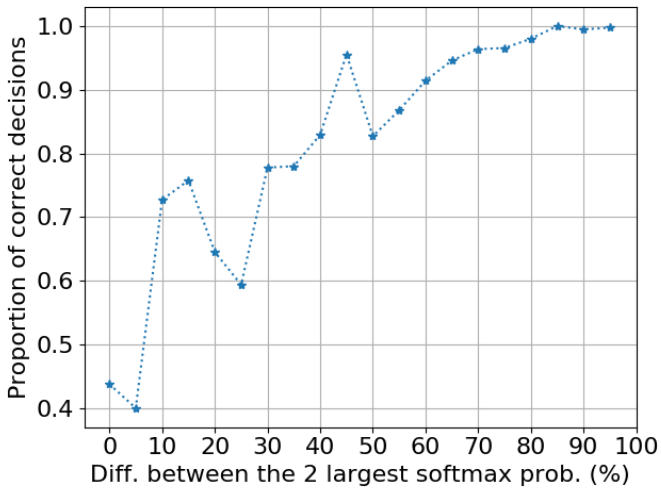


Figure 3.9: Relation between the proportion of correct decisions at classifier **(a)** (hierarchical classifier with non-shared features), in the validation set, and the difference between the two softmax probabilities.

As expected, the proportion of correct decisions increases as the difference between the two softmax probabilities increases.

We formulated an approach that combines the flat and hierarchical models, which we denote mixed model. With this model we aim to minimize the wrong decisions by the five classifiers belonging to the hierarchical model, by inspecting the value of the difference between the two largest softmax probabilities produced at each one.

During the inference stage, if at least one decision in the hierarchical model is not confident, the

image is sent to the flat model, otherwise it is classified with the hierarchical model.

To assess the confidence in each decision, we associate a hyper-parameter $\eta \in [0, 1]$ (or $[0, 100]\%$) to each classifier. We assume that a classifier is confident in its decision, if the difference between the two largest softmax probabilities is larger than its threshold. The five thresholds, η_i, are independent and may take different values. Figure 3.10 illustrates this methodology.

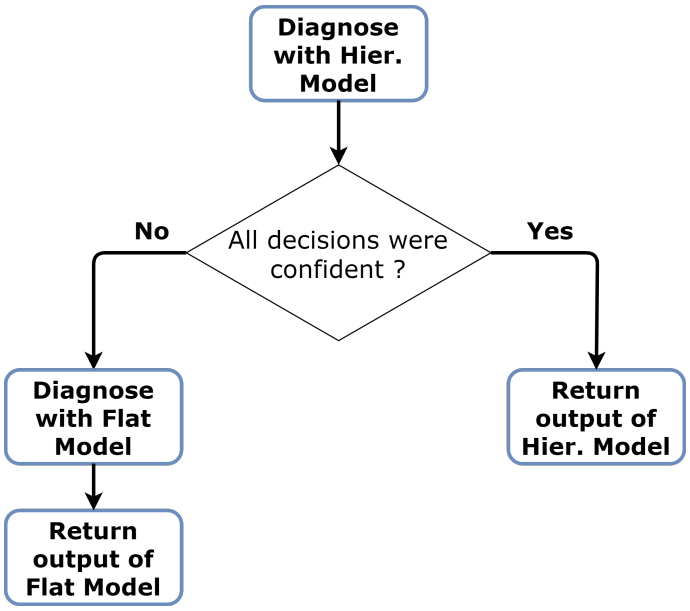


Figure 3.10: Steps to diagnose a skin lesion with the mixed model.

3.5 Techniques to Improve Performance

In order to improve the performance of the models described previously, same strategies were adopted. The most straightforward way to improve the performance of deep neural networks is by increasing their size (depth and width) [8]. However, this approach comes with a major drawback: bigger size typically means a larger number of parameters, which makes the enlarged network more prone to overfitting, especially if the number of labeled examples in the training set is limited [8]. A popular approach to prevent overfitting is to artificially augment the training set.

Also, the dataset, used in the experiments presented below, is very unbalanced, as we can see in figure 3.11. A popular approach to deal with an unbalanced dataset is to assign different weights to the classes in the cost function, as in [32]. We applied both techniques in our experiments.

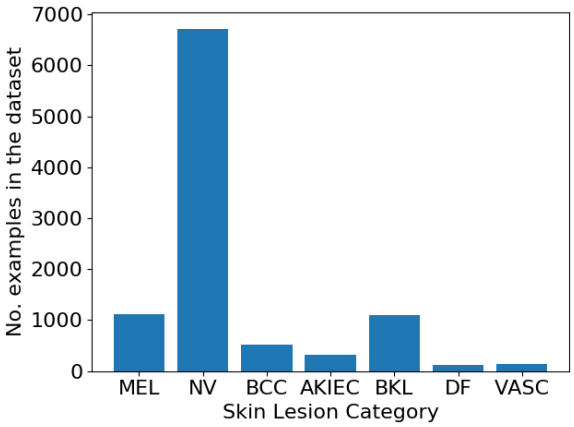
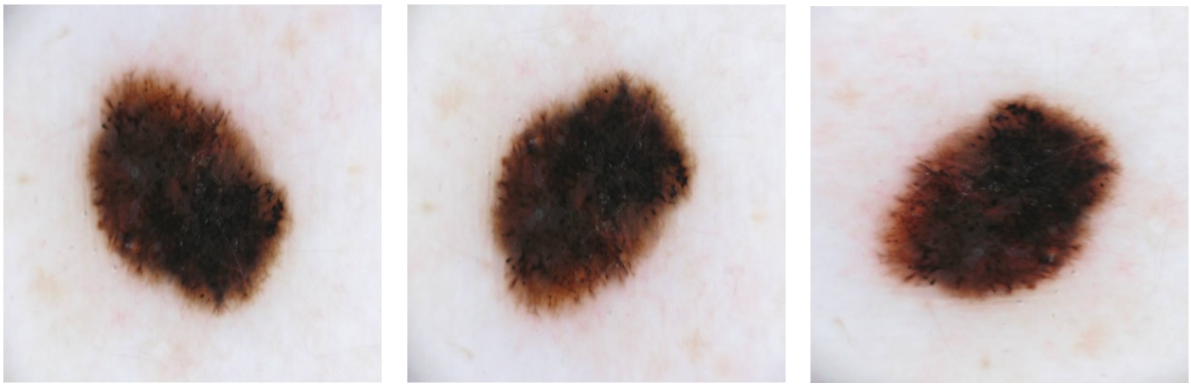


Figure 3.11: Distribution of the number of examples of each skin lesion category in the dataset used in this work.

3.5.1 Data Augmentation

In order to prevent overfitting, new artificial images were generated from the original ones, belonging to the training dataset. For each training image, two artificial images were generated, corresponding to the vertical flip and a 90º rotation (counter clockwise) of the original image (other data augmentation techniques could be used). This results in an augmented training set, triple the size of the original, which allows the utilization of a deep neural network with bigger depth and width. The figure 3.12 illustrates this process for one image.



(a) Original image.

(b) Vertical flip.

(c) Rotation of 90º (counter clockwise).

Figure 3.12: Data augmentation of a dermoscopy image.

3.5.2 Assigning a Unique Weight for each Category in the Loss Function

A popular technique used to deal with the unbalanced training set consists of assigning the class weights based on their distribution in the dataset:

$$w_c = \frac{N}{N_C}, \tag{3.2}$$

where N is the size of the training set and N_C is the number of training elements from class C. With this technique, the less frequent classes will be assigned to a higher weight and the loss becomes a weighted average. This technique prevents the model to be biased towards the most frequent categories, just because there are more training examples of this category in the dataset.

3.6 Dataset with Reduced Dimensions

Nowadays, it is very important to improve the efficiency of a classification system, with respect to the available data. In this work, we plan to study the effects of using a hierarchical model in a dataset with reduced dimensions.

The original dataset was split in half, and the same strategies and procedures were applied during this analysis (*i.e.*, data augmentation was performed with the same strategies, class weights were used and every training hyper-parameter was maintained).

The results were evaluated with the flat, hierarchical, and mixed models. Then, we compared the benefits of using these models in the original dataset and in the dataset with reduced dimensions.

4

Experimental Results

Contents

This chapter presents the experimental results and discussion regarding the benefits of using a hierarchical model to diagnose a skin lesion.

4.1 Dataset ISIC 2018

The dataset used in this work is the one released for the ISIC 2018 challenge [1]. It comprises 10,015 images with ground truth, used for training and validation, and 1,512 images without ground truth data (test set). The models' evaluation on the test set is performed on an online platform [46], with the hyper-parameters tuned in the validation set as described in chapter 3. This dataset comprises seven categories of skin lesions as shown in table 4.1.

Table 4.1: Number of examples of each skin lesion category in the dataset.

MEL	NV	BCC	AKIEC	BKL	DF	VASC	Total
1,113	6,705	514	327	1,099	115	142	10,015

This dataset was randomly divided in training ($\sim$ 80 %) and validation ($\sim$ 20 %) sets, resulting in a training set comprising 8,004 images and a validation set with 2,011 images.

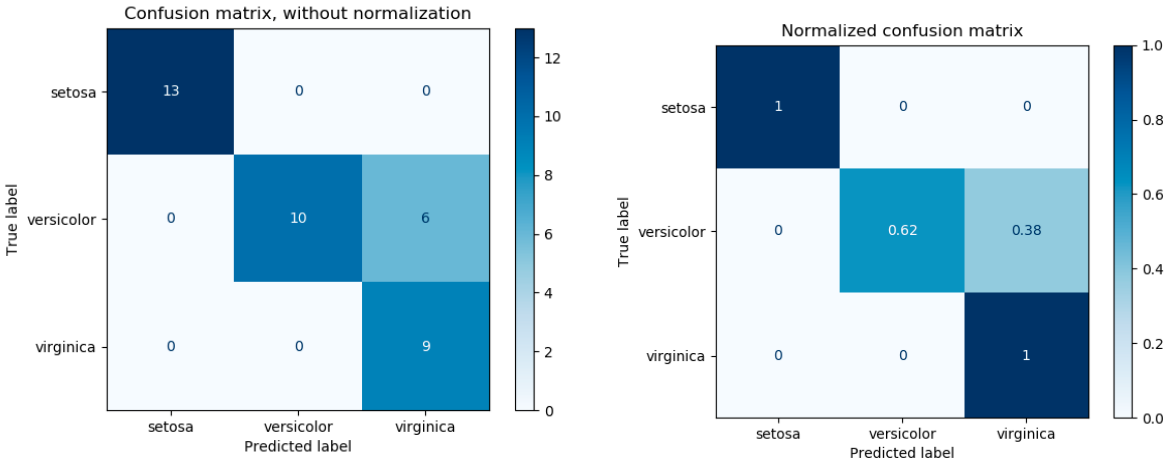
In order to prevent overfitting, data augmentation was performed on the training set, as described in section 3.5.1. The resulting augmented training set comprises 24,012 images. The distribution of the skin lesion categories in the augmented training and validation sets is presented in table 4.2.

Table 4.2: Number of examples of each skin lesion category in the augmented training and validation sets.

Dataset	MEL	NV	BCC	AKIEC	BKL	DF	VASC	Total
Train	2,697	15,936	1,272	834	2,625	291	357	24,012
Validation	214	1,393	90	49	224	18	23	2,011

4.2 Performance Metrics

The performance metrics are computed from the confusion matrix [47]. The confusion matrices shown in this work are normalized, meaning that they are obtained from the corresponding non-normalized confusion matrix, dividing the value of each element by the sum of the corresponding row elements. Figure 4.1 shows an example of a non-normalized and the corresponding normalized confusion matrix retrieved from [12].



(a) Non-normalized.

(b) Normalized.

Figure 4.1: Example of a non-normalized and corresponding normalized confusion matrices, retrieved from [12]

Each element P_{ij} of the normalized matrix P represents the probability $p(\hat{y} = j|y = i)$, where $\hat{y}$ and y are the predicted and the true labels, respectively; i and j are the row and columns of P, respectively. Our primary metric is the Balanced Accuracy (BACC) (the metric which will guide our decisions), which is the mean value of the diagonal elements in the normalized confusion matrix P:

$$\text{BACC} = \frac{1}{C}\sum_{i=1}^{C} p(\hat{y} = i|y = i) = \frac{1}{C}\sum_{i=1}^{C} P_{ii}, \tag{4.1}$$

where C is the number of different categories.

We will also compute the metric that we interpreted to be the primary metric used at the ISIC 2018 Challenge [1], which represents the probability of correctly predicting the category of a skin lesion. It is given by

$$P_{SUCC.} = \sum_{i=1}^{C} p(\hat{y} = i, y = i) = \sum_{i=1}^{C} p(\hat{y} = i|y = i) \cdot p(y = i) = \sum_{i=1}^{C} P_{ii} \cdot p(y = i). \tag{4.2}$$

where $p(y = i)$ is the probability of finding a skin lesion of class i in the evaluation set.

4.3 Computational Environment

All experiments were carried out using the programming language python 3.6, on a desktop computer with the following specifications:

- Processor: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz;
- Memory: 16 GB RAM;

- Storage: 1 TB HDD;
- Graphics Processing Unit (GPU): NVIDIA GeForce GTX 1060 6 GB.

All deep learning implementations are based on the open-source library Keras [48] on top of TensorFlow (v2.1.0) [49]. This framework computes back propagation and updates the models parameters automatically.

Appendix B presents the code for creating a flat and the hierarchical models with shared and non-shared features. It also presents the code for compiling, training and evaluating the flat model in the validation set, using the tensorflow framework [49].

4.4 Flat Classifier

As stated in section 3.2, we compared three different state-of-the-art CNN architectures: VGG-19, ResNet-101 and DenseNet-121. For each CNN we compared the results of training it from scratch and with transfer learning. In all experiments, the training was carried out using mini-batch with a size of 10 images, using the Adam optimizer, and the cross entropy loss function. The initial learning rate was set to 10^{-5}, and it was divided by 10 at 50 %, and 75 % of the total number of training epochs, as described in DenseNet paper [11]. The number of epochs was chosen using the validation set, *i.e.*, the model loaded the weights achieved in the epoch that led to the minimum cross entropy loss in the validation set.

Tables 4.3 and 4.4 show the performance of the CNNs trained from scratch and with transfer learning, respectively.

Table 4.3: Scores obtained with flat classifiers trained from scratch ordered from the best to the worst.

CNN	BACC	$P_{SUCC.}$
DenseNet-121	0.758	0.725
ResNet-101	0.664	0.652
VGG-19	0.634	0.633

Table 4.4: Scores obtained with flat classifiers trained with transfer learning ordered from the best to the worst.

CNN	BACC	$P_{SUCC.}$
DenseNet-121	0.810	0.826
ResNet-101	0.786	0.809
VGG-19	0.690	0.708

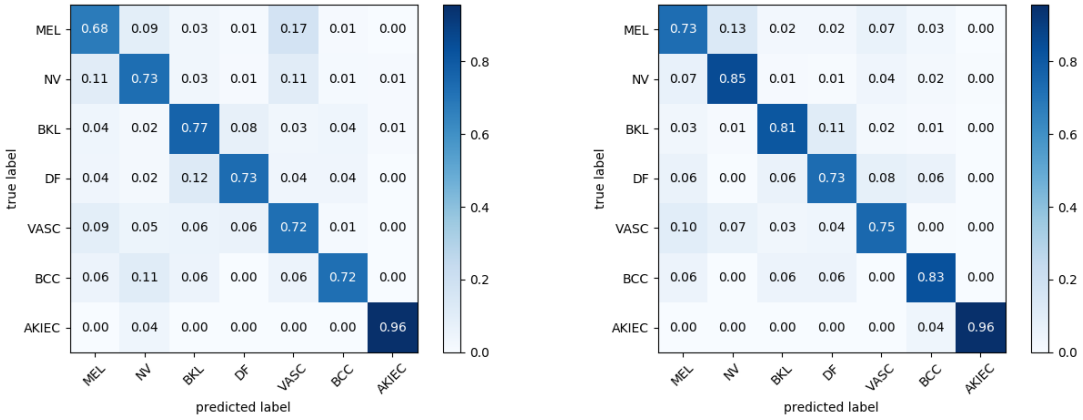
These results show that for every CNN, the performance is better when it is trained with transfer learning, as expected . Also, independently of training the CNNs from scratch or with transfer learning,

DenseNet-121 is the CNN which achieves the best performance, followed by ResNet-101. There is a significant difference between the performances of DenseNet-121 and the other CNNs, when they are trained from scratch.

The following sections present the detailed results obtained with DenseNet-121, which achieved the best performance. The results obtained with VGG-19 and ResNet-101 are presented in Appendix A.

4.4.1 DenseNet-121

Figure 4.2 presents the confusion matrices obtained with the DenseNet-121 model in the validation set, trained from scratch and with transfer learning.



(a) DenseNet-121 trained from scratch.

(b) DenseNet-121 trained with transfer learning.

Figure 4.2: Confusion matrices of DenseNet-121 obtained in validation set.

When analysing both confusion matrices we observe the following for each category:

$$p(\hat{y} = i|y = i)|_{TL} \geq p(\hat{y} = i|y = i)|_{SC}, \qquad i \in \{1, ..., 7\}, \tag{4.3}$$

where TL and SC refer to the confusion matrices obtained with transfer learning and from scratch, respectively. This means that transfer learning did not degrade the performance in any category. Also, the scores of NV and BCC improved significantly when the model was trained with transfer learning,

4.5 Hierarchical Model

4.5.1 Non-Shared Features

As described in section 3.3.1, this approach trains a different model for each decision in the hierarchy. Two hierarchical models were implemented, following this approach: (i) using ResNet-101, and (ii) using DenseNet-121. All CNNs were trained using the same configurations and strategies as in the training of the flat classifiers with transfer learning.

Table 4.5 presents the scores obtained with the hierarchical classifiers with non-shared features.

Table 4.5: Scores obtained with the hierarchical classifiers with non-shared features.

Hierarchical	BACC	$P_{SUCC.}$
DenseNet-121	0.773	0.825
ResNet-101	0.747	0.775

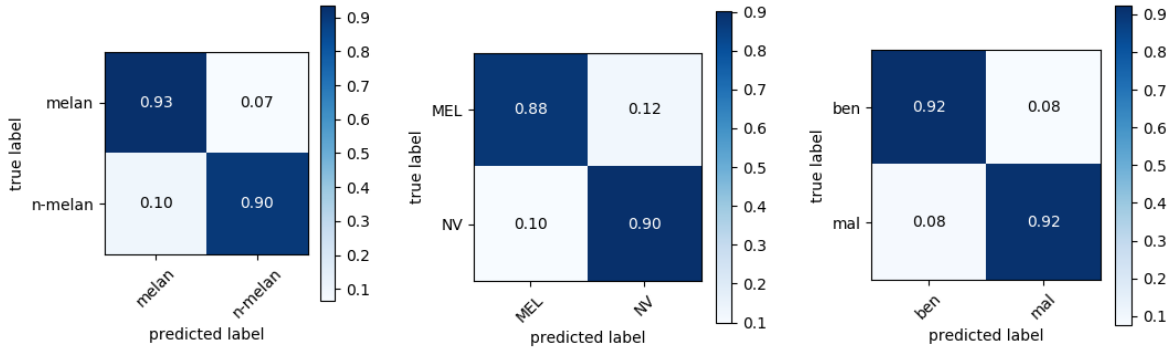
Similarly to the results of the flat classifier, the hierarchical network implemented with DenseNet-121 achieved better results. Next, we present the detailed results obtained with the hierarchical model with non-shared features, implemented with DenseNet-121.

Figure 4.3 shows the confusion matrices of each of the individual classifiers that belong to the hierarchical model with non-shared features, implemented with DenseNet-121. In the confusion matrices presented, only images that correctly arrive at the classifier are counted. For instance, if a melanocytic lesion arrives at classifier **(c)**, which should only receive non-melancocytic lesions, this prediction will not be saved in the confusion matrix of **(c)**.

When comparing the diagonal elements of these confusion matrices, we observe that all of them present values around 90 %, and the worst score is $p(\hat{y} = \text{BCC}|y = \text{BCC})$, at classifier **(e)**.

The intermediate decisions are performed with the classifiers **(a)** and **(c)**, which predict incorrectly the classes of 144 and 29 images, respectively, resulting in a total of 173 incorrect intermediate decisions. The final decisions are performed with the classifiers **(b)**, **(d)**, and **(e)**, which predict incorrectly the classes of 153, 9, and 16 images, resulting in a total of 178 incorrect final decisions. The incorrect intermediate decisions are slightly less than the incorrect final decisions, and are contributing to reduce the performance of the hierarchical model.

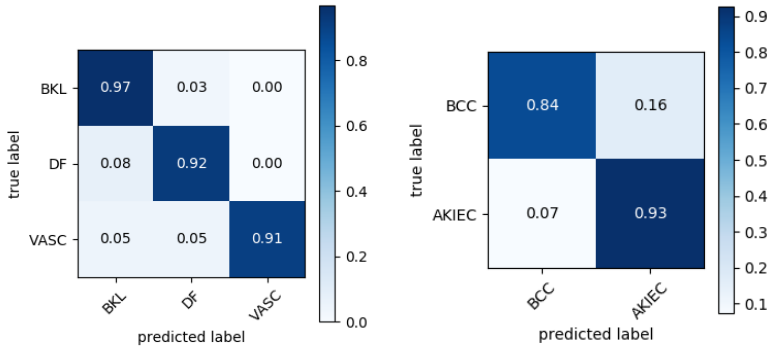
By evaluating the model at the validation set with a hierarchical inference procedure, we obtain the confusion matrix presented in figure 4.4.



(a) Classifier **(a)** of melanocytic vs. non-melanocytic lesions.

(b) Final classifier of **(b)** **MEL** and **NV** lesions.

(c) Classifier **(c)** of benign vs. malignant non-melanocytic lesions.



(d) Final classifier **(d)** of **BKL**, **DF** and **VASC**.

(e) Final classifier **(e)** of **BCC** and **AKIEC** lesions.

Figure 4.3: Confusion matrices of individual CNNs belonging to the hierarchical classifier with non-shared features, implemented with DenseNet-121, evaluated in the validation set.

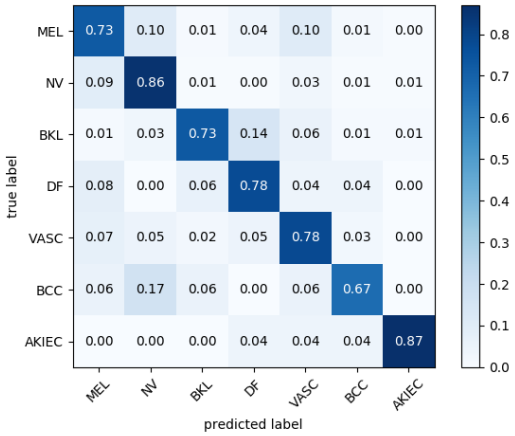


Figure 4.4: Confusion matrix obtained with the hierarchical classifier with non-shared features, implemented with DenseNet-121.

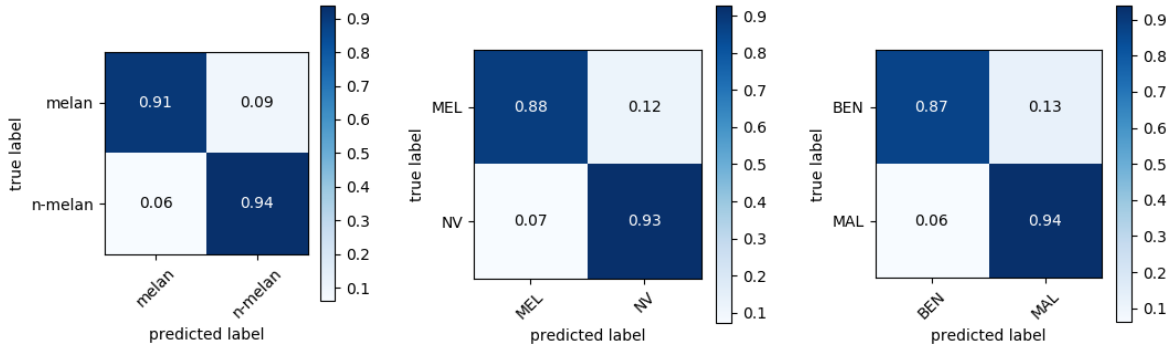
The category i with the worst $p(\hat{y} = i|y = i)$ is BCC. On the other hand, NV and AKIEC are the categories with the best $p(\hat{y} = i|y = i)$. When compared to the best model of the flat approach, this method degraded the scores of BKL, BCC, and AKIEC, and the improvements at the other categories were not so significant, resulting in a worse BACC.

4.5.2 Shared Features

Figure 4.5 shows the confusion matrices of each of the individual classifiers that belong to the hierarchical model with shared features, implemented with DenseNet-121. As before, these confusion matrices only account for images that correctly arrive at the classifier.

Except for the classifier **(e)**, the classifiers trained in this approach present similar results, with an average of the diagonal values greater than 90 %. Similarly to the hierarchical model with non-shared features, the classifier **(e)**, which predicts the final category of non-melanocytic malignant lesions, is the classifier with the worse results. This may be caused by the fact that non-melanocytic malignant lesions are the categories less represented in the dataset.

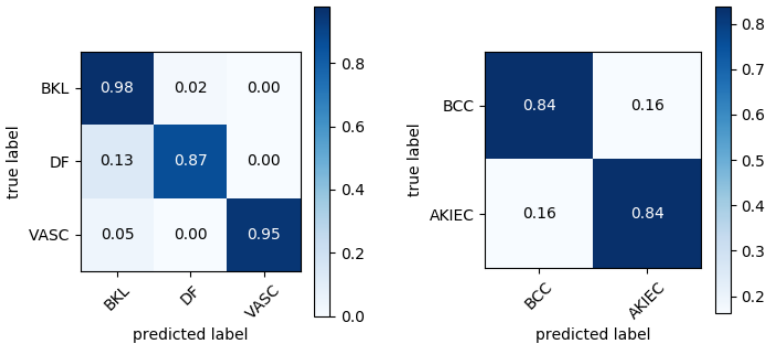
The intermediate classifiers, **(a)** and **(c)**, predict incorrectly the classes of 171 and 32 images, respectively, resulting in a total of 203 incorrect intermediate decisions. The final decisions are performed with the classifiers **(b)**, **(d)**, and **(e)**, which predict incorrectly the classes of 115, 7, and 19 images, resulting in a total of 141 incorrect final decisions. This behaviour shows that the hierarchical model with shared features is more penalized with the incorrect intermediate decisions than the hierarchical model with non-shared features. This behaviour confirms the importance of minimizing these incorrect intermediate decisions.



(a) Classifier **(a)** of melanocytic vs. non-melanocytic lesions.

(b) Final classifier of **(b)** **MEL** and **NV** lesions.

(c) Classifier **(c)** of benign vs. malignant non-melanocytic lesions.



(d) Final classifier **(d)** of **BKL**, **DF** and **VASC**.

(e) Final classifier **(e)** of **BCC** and **AKIEC** lesions.

Figure 4.5: Confusion matrices of individual CNNs belonging to the hierarchical classifier with shared features, implemented with DenseNet-121, evaluated in the validation set.

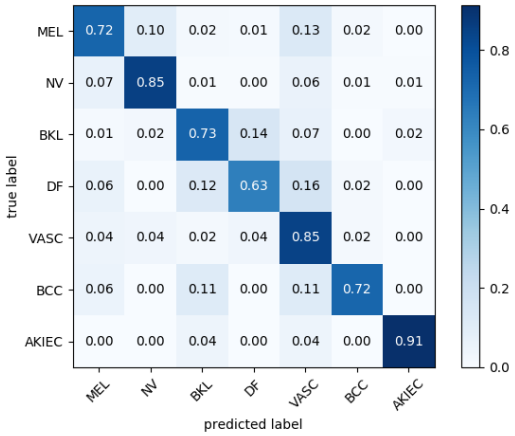


Figure 4.6: Confusion matrix obtained with the hierarchical classifier with a single CNN as feature extractor, implemented with DenseNet-121.

Figure 4.6 presents the confusion matrix obtained with this approach. This model achieved a BACC $= 0.776$ and a $P_{SUCC.} = 0.829$. When comparing this confusion matrix with the one obtained with the hierarchical model using non-shared features, we observe that there is an improvement at the categories VASC, BCC, and AKIEC. However, there is a significant deterioration at the category DF.

4.6 Comparison of Flat and Hierarchical Models' Results

Table 4.6 compares the performance obtained in the validation set, with the hierarchical and flat models implemented with DenseNet-121.

Table 4.6: Comparison of the performance obtained with the hierarchical and flat classifiers, implemented with DenseNet-121.

	BACC	$P_{SUCC.}$	No. parameters
Flat	0.810	0.826	7,044,679
Hierarchical - shared	0.776	0.829	7,048,779
Hierarchical - non-shared	0.773	0.825	35,198,795

The flat classifier achieves a better BACC than any of the hierarchical models, and the primary metric used at the ISIC 2018 Challenge, $P_{SUCC.}$, is similar among the models. The performances of the hierarchical models are similar but the hierarchical model with shared features among the classifiers requires less resources to train and store the model's weights.

The final decisions of a hierarchical classifier are performed in the last level of the hierarchy, however, the errors produced in previous levels of decision are propagated to the subsequent levels and prevent the subsequent decisions from being correct. This error propagation phenomenon caused a decrease

in the performance of the hierarchical models.

4.7 Mixed Model - Hierarchical with Non-shared Features

In order to reduce the number of predictions affected by the error propagation phenomenon in the hierarchical model, a mixed model of a flat and a hierarchical classifier was implemented, as described in section 3.4. Since both hierarchical and flat classifiers implemented with DenseNet-121 outperformed the ones implemented with other state-of-the-art CNNs, the mixed model was implemented with DenseNet-121. In this section we present the results obtained with the mixed model using the best flat classifier and the hierarchical model with non-shared features.

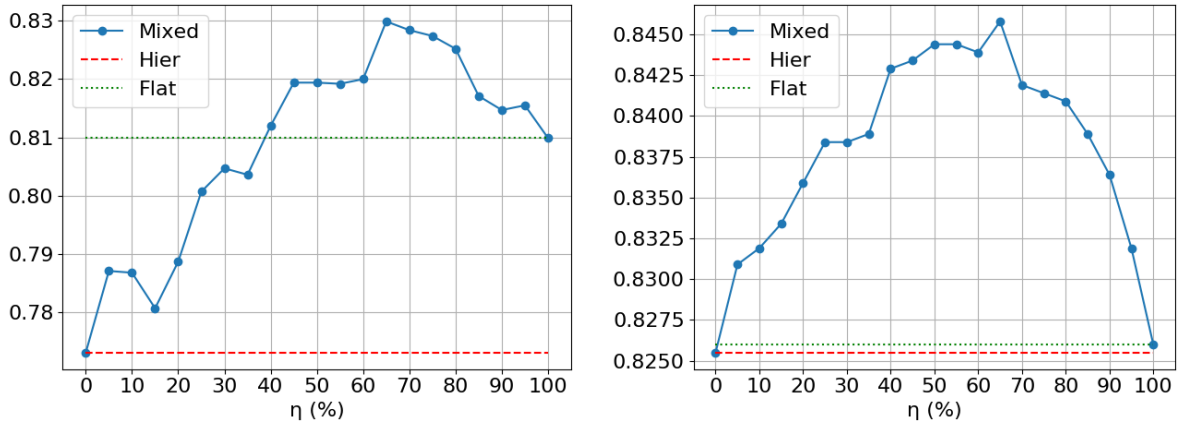
As stated in section 3.4, during the inference stage, we assess if for any of the classifiers' outputs the difference between the two largest softmax probabilities is less than a threshold. For the positive cases, the image is sent to the flat model, otherwise it is classified with the hierarchical model. We denote $\vec{\eta} = (\eta_a, \eta_b, \eta_c, \eta_d, \eta_e)$, whose elements correspond to the threshold values from classifiers **(a)** to **(e)**. The hyper-parameters η_i were tuned using the validation set.

4.7.1 Same η for the Five Classifiers

In this approach, the classifiers are assumed to share the same threshold η. The model was evaluated in the validation set, by varying this hyper-parameter from 0 % (hierarchical classifier) to 100 % (flat classifier) with a step of 5 %. For each η, the BACC and the $P_{SUCC.}$ were annotated. In figure 4.7, the scores are reported for different values of η.

We observe for both metrics that initially, by increasing η, the performance has an upward behaviour. After reaching the maximum value, performance starts to degrade. After this value, the classifiers of the hierarchical model are confident enough and it is not advantageous to send the image to the flat classifier. We observe that the mixed model performs better than the flat classifier, in terms of BACC, only if $\eta \geq 40\%$. In terms of $P_{SUCC.}$, the mixed model outperforms the other methods independently of the chosen η.

In figure 4.8 it is represented the number of images that were forwarded to the flat classifier for each value of η.



(a) BACC. **(b)** $P_{SUCC.}$.

Figure 4.7: Metrics obtained with the mixed classifier in function of the hyper-parameter η.

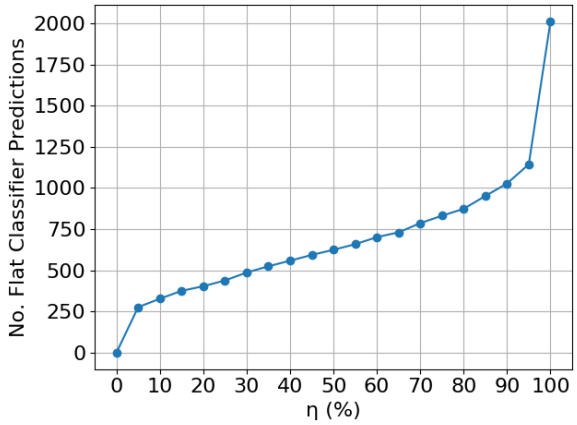


Figure 4.8: Number of flat classifier's predictions in function of η.

As expected, the curve is increasing monotonic. When $\eta = 0$, the mixed model is equivalent to the hierarchical model because no image is forwarded to the flat classifier. On the contrary, when it is equal to 100 %, we have a flat classifier because all the images are forwarded to the flat classifier.

As seen in figure 4.7, the highest BACC is achieved with $\eta = 65\%$. The confusion matrix obtained with this mixed model, evaluated in the validation set, is presented in figure 4.9.

When compared to the flat approach, which performed better than the hierarchical approach, the mixed model improved significantly the score of DF, slightly degraded the score of AKIEC and slightly improved the scores of the other categories.

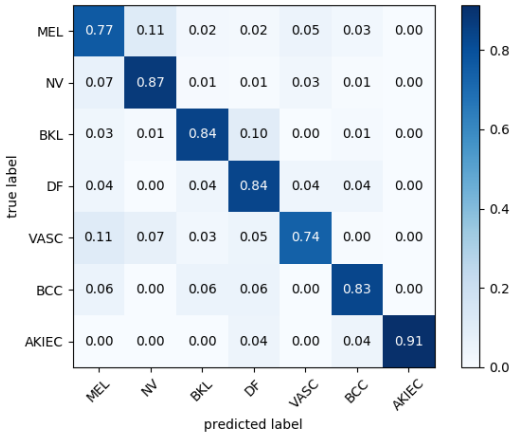


Figure 4.9: Confusion matrix obtained with the mixed model, using $\eta = 65\%$.

Table 4.7 summarizes the performances obtained with the different models.

Table 4.7: Comparison of the performance obtained with the hierarchical, flat and mixed models, implemented with DenseNet-121, from the best to the worst. Both hierarchical and mixed models are using the hierarchical model with non-shared features.

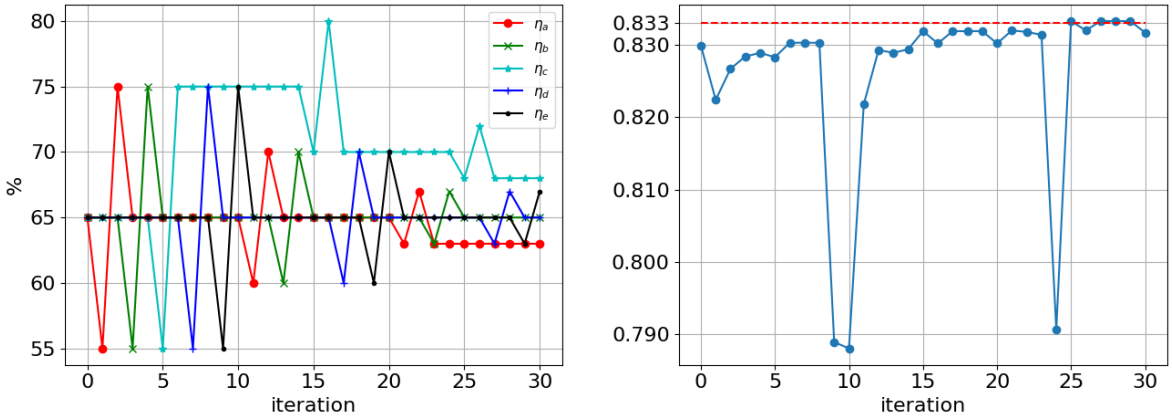
DenseNet-121	BACC	$P_{SUCC.}$
Mixed($\vec{\eta} = (65, 65, 65, 65, 65)\%$)	0.830	0.846
Flat	0.810	0.826
Hierarchical	0.773	0.825

4.7.2 Independent η for each of the Five Classifiers

Since each classifier has a different classification task, it is expected that the optimal value of η_i may be different among the classifiers. Our method for chosing and evaluating different combinations of $\vec{\eta} = (\eta_a, \eta_b, \eta_c, \eta_d, \eta_e)$ is the following:

1. Initialize $\vec{\eta}$ with the best combination obtained on section 4.7.1, with the same η for the five classifiers. Initialize $\vec{\Delta}$ which defines the how each η_i will be changed.
2. Pick the next value of $\vec{\Delta}$.
3. Evaluate the model with $\vec{\eta} = (\eta_a - \Delta, \eta_b, \eta_c, \eta_d, \eta_e)$ and with $\vec{\eta} = (\eta_a + \Delta, \eta_b, \eta_c, \eta_d, \eta_e)$.
4. Update $\vec{\eta}$ with the best combination among $\vec{\eta} = (\eta_a - \Delta, \eta_b, \eta_c, \eta_d, \eta_e)$, $\vec{\eta} = (\eta_a, \eta_b, \eta_c, \eta_d, \eta_e)$, and with $\vec{\eta} = (\eta_a + \Delta, \eta_b, \eta_c, \eta_d, \eta_e)$
5. Repeat from step 3. but now changing the next η_i, until every η_i update is tested.
6. Repeat from step 2. with the next value of $\vec{\Delta}$, until all values are tested.

By following this method, we initialized $\vec{\eta} = (65, 65, 65, 65, 65)$ % which was the best combination obtained in the previous approach, and initialized $\vec{\Delta} = (10, 5, 2)$ %. Figure 4.10 shows the values of $\vec{\eta}$ in each iteration as well as the corresponding BACC. Each iteration represents a different combination of the values of $\vec{\eta}$, that was tested. The best configuration is achieved at iteration 25, with $\vec{\eta} = (63, 65, 68, 65, 65)$ %, with BACC $= 83.3\%$



(a) $\vec{\eta}$ in each iteration.

(b) BACC obtained in each iteration.

Figure 4.10: Combinations of η_i and BACC in each iteration.

This method for evaluating different combinations of $\vec{\eta} = (\eta_a, \eta_b, \eta_c, \eta_d, \eta_e)$ does not improve the performance of the model significantly, when compared to the one obtained with the same η for the five classifiers.

Also, the method for tuning the values of each η_i is robust, since it always outputs a $\vec{\eta}$ that achieves an increased or equal BACC than the initial $\vec{\eta}$. However, in iterations 9, 10 and 24 the BACC significantly decreases. Iterations 9 and 10 are related to changes of -10 and $+10$ in η_e, respectively. Iteration 24 corresponds to a change of $+2$ in η_b.

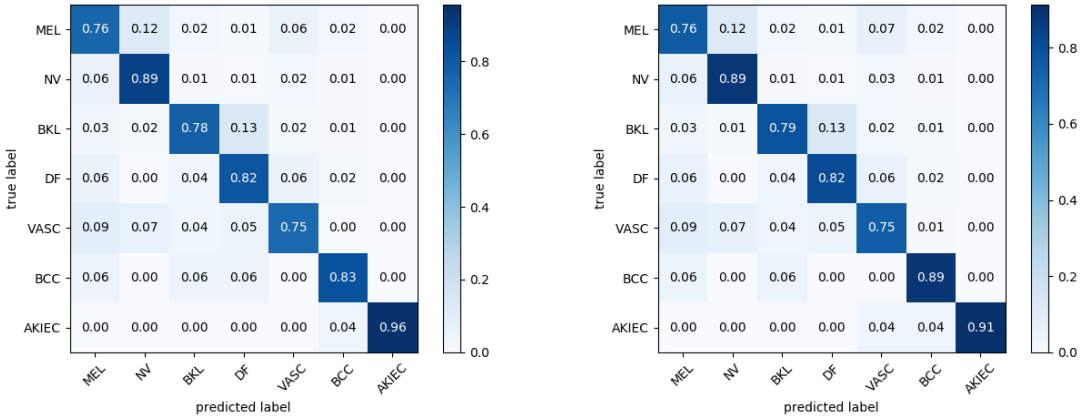
4.8 Mixed Model - Hierarchical with Shared Features

The same procedures were adopted to develop a mixed model, using the hierarchical model with a single CNN as feature extractor. The best configuration achieved with the same η in the five classifiers was $\vec{\eta} = (55, 55, 55, 55, 55)$ %, resulting in a BACC $= 0.826$ and a $P_{SUCC.} = 0.853$.

Then we applied the strategy described in section 4.7.2 to evaluate different combinations of $\vec{\eta}$ with different η_i, using $\vec{\Delta} = (10, 5, 2)$ %. The best configuration obtained was $\vec{\eta} = (65, 55, 55, 40, 55)$, achieving a BACC $= 0.829$ and a $P_{SUCC.} = 0.852$, which once again, does not improve the performance

of the model significantly, when compared to the one obtained with the same η for the five classifiers. This combination of η_i changed the values of η_a and η_d and maintained the others.

The confusion matrices are presented in figure 4.11.



(a) Best configuration with the same η among the five classifiers. $\vec{\eta} = (55, 55, 55, 55, 55)$.

(b) Best configuration with different η_i among the five classifiers. $\vec{\eta} = (65, 55, 55, 40, 55)$.

Figure 4.11: Confusion matrices of the mixed model, using a hierarchical model with one CNN.

These matrices present similar results. By changing the values of η_a and η_d, the model improved the scores of BKL and BCC, and degraded the score of AKIEC.

The mixed approach comprising a hierarchical model with shared features, using $\vec{\eta} = (65, 55, 55, 40, 55)$, degraded the flat model's scores of BKL and AKIEC, but improved the scores of MEL, NV, DF, and BCC, achieving a better BACC.

When comparing it with the hierarchical model with shared features, we observe that it only degraded the score of VASC, improving or maintaining the other scores.

4.9 Experiments in Dataset with Reduced Dimensions

The medical images necessary for the development of these methods are often not available. This is more and more frequent given the strict data protection policy. Additionally, many algorithms (especially, supervised learning algorithms) require labels created by doctors who have other and more important tasks and responsibilities. Therefore, there is a need to develop algorithms that are robust in the presence of little data. In this work we evaluate the robustness of the implemented models with respect to the amount of the available data, by comparing the models' results obtained when trained in the original dataset and in a new dataset, about half the size.

The dataset released for the ISIC 2018 challenge comprises 10,015 images with ground truth. Each image of this dataset was included in a new training set with probability of 40 %, and in a new validation set with a probability of 10 % (each image could only belong at most to one set). Finally, we applied data augmentation to the training set, as described in section 3.5.1.

The number of examples of each skin lesion category in the training and validation sets, is presented in table 4.8 .

Table 4.8: Number of examples of each skin lesion category in the reduced training and validation sets.

Dataset	MEL	NV	BCC	AKIEC	BKL	DF	VASC	Total
Train	1,212	7,149	519	360	1,101	114	153	10,608
Validation	94	621	40	24	94	7	12	892

First, we used three different state-of-the-art CNN architectures, to implement a flat model: VGG-19, ResNet-101, and DenseNet-121, obtaining the scores presented in figure 4.9.

Table 4.9: Metrics obtained with flat classifiers, in reduced validation set, trained with transfer learning, ordered from the best to the worst.

CNN	BACC	$P_{SUCC.}$
DenseNet-121	0.734	0.748
ResNet-101	0.659	0.648
VGG-19	0.636	0.656

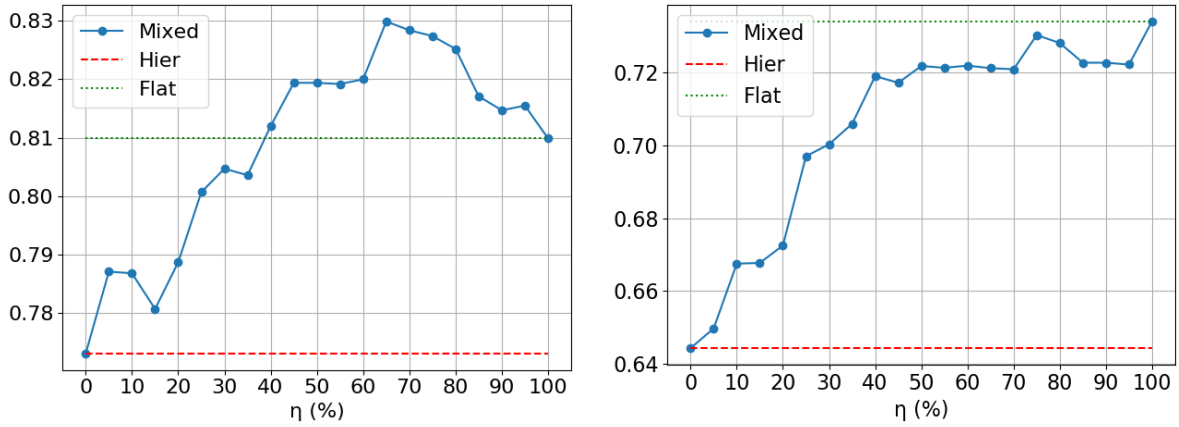
The DenseNet-121 architecture continues to outperform the others, and all the hierarchical and mixed models were implemented with this CNN.

Table 4.10 presents the scores obtained with the hierarchical models implemented with shared and non-shared features.

Table 4.10: Metrics obtained with the hierarchical models, in reduced validation set.

	BACC	$P_{SUCC.}$
Non-shared Features	0.644	0.743
Shared Features	0.537	0.638

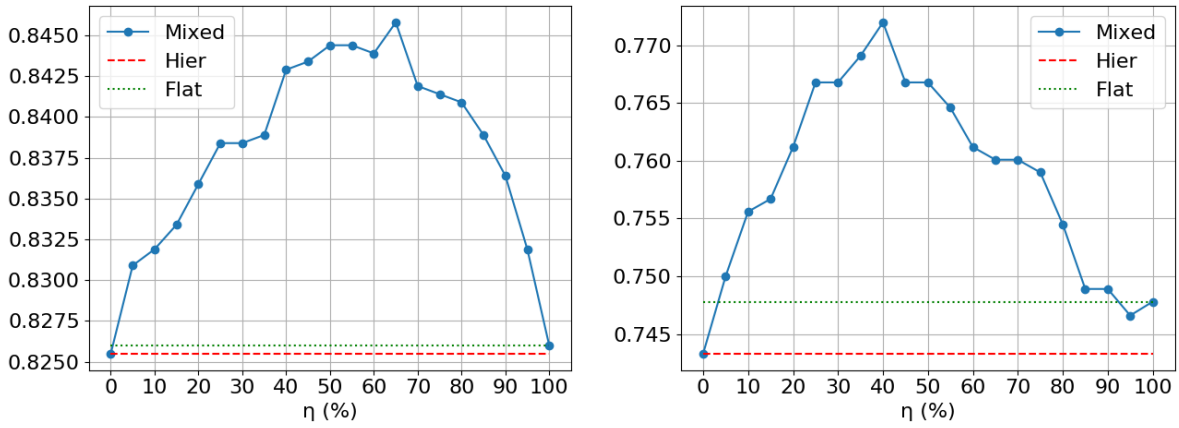
A mixed model was implemented, using the hierarchical model with the best results (with non-shared features). The η threshold was varied from 0 % to 100 % with a step of 5 %, but none achieved a greater BACC than the one obtained with the flat model, as shown in figure 4.12. The P_{SUCC} obtained with the mixed models is shown in figure 4.13, and the maximum value of $P_{SUCC.}$ is obtained with $\eta = 40\%$.



(a) Original dataset.

(b) Reduced dataset.

Figure 4.12: BACC obtained with the flat, hierarchical and mixed models with the same η among the classifiers, in function of η, in datasets with different dimensions.

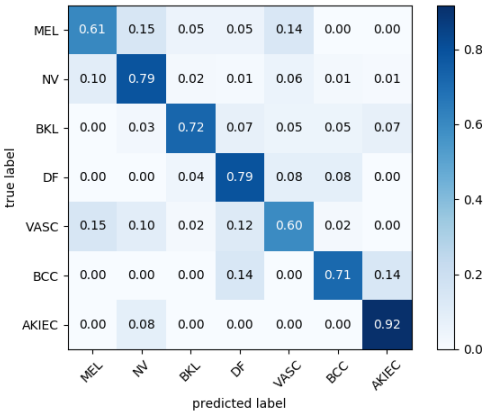


(a) Original dataset.

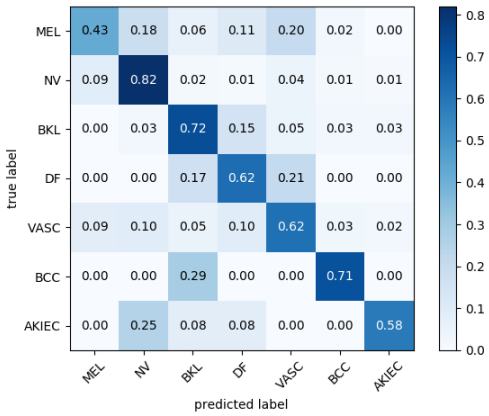
(b) Reduced dataset.

Figure 4.13: $P_{SUCC.}$ obtained with the flat, hierarchical and mixed models with the same η among the classifiers, in function of η, in datasets with different dimensions.

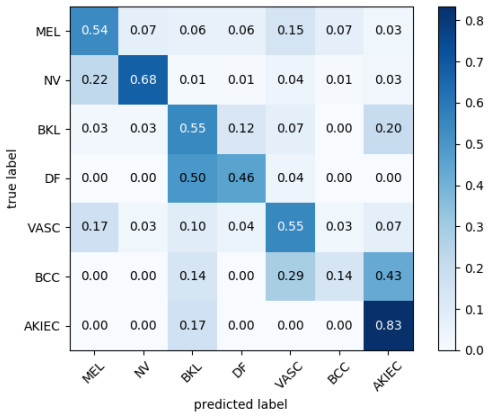
Figure 4.14 present the confusion matrices obtained with the best flat model (using DenseNet-121), and the hierarchical models with non-shared and shared features.



(a) Flat model using DenseNet-121.



(b) Hierarchical model with non-shared features.



(c) Hierarchical model with shared features.

Figure 4.14: Confusion matrices of flat and hierarchical models trained in the small training set, and evaluated in the small validation set.

As expected, the scores obtained with the models in the small dataset were worse than the ones obtained in the original dataset.

In the small dataset, the performance of the flat model is significantly better than the hierarchical models' performances. Regarding the flat model, the two best categories (NV and AKIEC) and the two worse categories (MEL and VASC) were preserved in both datasets. Except for the category DF, all the remaining categories present a better score in the original dataset.

The hierarchical model with non-shared features compared to the flat model, slightly improved the

scores of categories NV and VASC, but degraded or preserved the scores of the remaining categories, achieving a worse BACC.

The hierarchical model with shared features presents the worse results. Many BCC lesions were predicted as AKIEC lesions, which are also non-melanocytic malignant lesions. The same behaviour happened with DF lesions, which were many times diagnosed as BKL lesions.

Since the validation set in this experiment is different from the validation set used throughout this work, the results obtained with the different approaches in both datasets are not fairly comparable. Thus, in the next section we present the results obtained in the test set, and compare the different approaches' results.

4.10 Final Evaluation in the Test Set

The test set is an independent set provided without ground truth data, and the evaluation of the models was performed on an online platform [46]. Table 4.11 summarizes the performance achieved by the models implemented with DenseNet-121 in the validation and held-out test set. The table is ordered from the best to the worst model in terms of BACC in the validation set.

Table 4.11: Global comparison of the performance achieved by the models on validation and test sets. The performances obtained with mixed and hierarchical model are identified with "non-shared" or "shared", if the hierarchical is implemented with non-shared or shared features, respectively. The score on the test set is obtained after submitting the model in the online platform.

Model	BACC (val.)	$P_{SUCC.}$ (val.)	Score (test)
Mixed-non-shared($\vec{\eta} = (63, 65, 68, 65, 65)\%$)	0.833	0.846	0.740
Mixed-non-shared($\vec{\eta} = (65, 65, 65, 65, 65)\%$)	0.830	0.846	0.737
Mixed-shared($\vec{\eta} = (65, 55, 55, 40, 55)\%$)	0.829	0.852	0.730
Mixed-shared($\vec{\eta} = (55, 55, 55, 55, 55)\%$)	0.826	0.853	0.730
Flat	0.810	0.826	0.712
Hier-shared	0.776	0.829	0.695
Hier-non-shared	0.773	0.825	0.713

The flat model performs better than the hierchical models in the validation set. Regarding the test set, it performs better than the hierarchical model with shared features, and presents similar results to those of the hierarchical model with non-shared features. Such behavior may be due to the fact that a pure hierarchical network is penalized with the wrong decisions of levels prior to the final decision.

With respect to the hierarchical models, the one with a single CNN as feature extractor performed slightly better in the validation set, but it achieved the worst performance in the test set, which suggests a worst capacity of generalization.

The introduction of a criterion that allows forwarding the image to the flat classifier, when the hierarchical network is not sufficiently confident, allowed the implementation of a mixed model, which achieved

better performance when compared to the corresponding flat and hierarchical models.

Except for the hierarchical model with shared features, the scores obtained in the test set present a similar behaviour as the scores obtained in the validation set, which means that the models are capable of generalization, even when tuning hyper-parameters (η).

Table 4.12 presents the results obtained in the test set, when the flat and hierarchical models are trained on the reduced set. A relative deviation is computed in order to assess the performance decrease in each approach.

Table 4.12: Comparison of the scores achieved by the flat and hierarchical approaches on the test set, with the models trained and validated in the original dataset or in a dataset with about half the size.

Approach	Score (original)	Score (reduced)	Relative Deviation
Flat	0.712	0.709	- 0.421 %
Hier-non-shared	0.713	0.629	- 11.8%
Hier-shared	0.695	0.461	- 33.7 %

We conclude that reducing the size of the dataset affected mostly the hierarchical models, in particular the hierarchical model with shared features. On the other hand, the flat approach almost maintained its performance, which is impressive since it was trained and validated in a halved dataset. The hierarchical approaches require enough examples of each class in order to capture the hierarchical structure of the dataset. If this is not possible, the flat approach may be the preferred one.

5

Conclusions and Future Work

Contents

5.1 Conclusions

The main goal of this thesis was to compare the performance of flat and hierarchical models, regarding the classification of skin lesion images. Also, we wanted to assess the effects of using a hierarchical model in a smaller dataset. The objectives have been accomplished.

Regarding the classification of dermoscopy images, flat classifiers do not take advantage of their hierarchical structure and the classification relies on a single decision including all the final categories.

On the other hand, a hierarchical model takes advantage of the hierarchical structure of skin lesions but suffer from the fact that they have to make decisions prior to reach a final category. This intermediate decisions lead to the error propagation phenomenon causing a decrease in accuracy when compared to the one obtained by the flat model. Moreover, a hierarchical classifier performs better in larger datasets.

In order to capture the strengths of both approaches, a mixed model was implemented, which followed a criterion to forward the dermosocopy image to the flat classifier, when the hierarchical classifier was not confident enough. This model allowed to improve the performance, when comparing to the flat and hierarchical results. Moreover, this approach presented good results and a good generalization capability, even when fine-tuning some hyper-parameters in the validation set.

Finally, we compared three state-of-the-art CNNs: VGG-19, ResNet-101, and DenseNet-121. DenseNet-121 outperformed the others in all the experiments. In DenseNet, each layer receives directly all the feature maps obtained with the preceding layers. This provides a strong gradient flow and improves the features' propagation to the last layer.

5.2 Future Work

As a follow-up to this work, some aspects can be studied. For instance, we provide a few examples:

- Structure the dataset using a different hierarchical organization with the intermediate decision levels in the following order: (i) benign vs. malignant, and (ii) melanocytic vs. non-melanocytic, instead of (i) melanocytic vs. non-melanocytic, and (ii) benign vs. malignant.
- The dataset used in this work only comprised a single category for melanocytic benign lesions, as well as for melanocytic malignant lesions, which allowed us to omit the decision level melanocytic vs. non-melanocytic. We think that should be interesting to assess the effects of using a hierarchical model in a dataset that specified the different categories of benign melanocytic lesions.

Bibliography

[1] N. Codella, V. Rotemberg, P. Tschandl, M. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, H. Kittler, and A. Halpern, "Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)," CoRR, vol. abs/1902.0, 2 2019. [Online]. Available: http://arxiv.org/abs/1902.03368

[2] "Task 3: Lesion Diagnosis — ISIC 2018," 2020. [Online]. Available: https://challenge2018.isic-archive.com/task3/

[3] "CS231n Convolutional Neural Networks for Visual Recognition," 2020. [Online]. Available: https://cs231n.github.io/neural-networks-1/

[4] "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges," 2020. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[5] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 and CIFAR-100 datasets," 2020. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[7] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Neural Information Processing Systems, vol. 25, 2012.

[8] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in 3rd International Conference on Learning Representations, ICLR 2015, San

Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," CoRR, vol. abs/1512.0, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[11] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," CoRR, vol. abs/1608.0, 8 2016. [Online]. Available: https://arxiv.org/abs/1608.06993

[12] "Confusion matrix — scikit-learn 0.23.1 documentation," 2020. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

[13] N. Codella, D. Gutman, M. Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kalloo, K. Liopyris, N. Mishra, H. Kittler, and A. Halpern, "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)," CoRR, vol. abs/1710.0, 10 2017. [Online]. Available: http://arxiv.org/abs/1710.05006

[14] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," CoRR, vol. abs/1811.0, 11 2018. [Online]. Available: http://arxiv.org/abs/1811.03378

[15] "Ultraviolet (UV) radiation and skin cancer," 2020. [Online]. Available: https://www.who.int/news-room/q-a-detail/ultraviolet-(uv)-radiation-and-skin-cancer

[16] "Skin Cancer Facts & Statistics - The Skin Cancer Foundation," 2020. [Online]. Available: https://www.skincancer.org/skin-cancer-information/skin-cancer-facts/

[17] K. M. Li and E. C. Li, "Skin Lesion Analysis Towards Melanoma Detection via End-to-end Deep Learning of Convolutional Neural Networks," CoRR, vol. abs/1807.0, 7 2018. [Online]. Available: http://arxiv.org/abs/1807.08332

[18] 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," CoRR, vol. abs/1409.0, 9 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[19] G. Argenziano and H. P. Soyer, Interactive atlas of dermoscopy. Edra Medical Publishing & New Media, 2000.

[20] H. Kittler, H. Pehamberger, K. Wolff, and M. Binder, "Diagnostic accuracy of dermoscopy," The Lancet Oncology, vol. 3, no. 3, pp. 159–165, 2002. [Online]. Available: https://doi.org/10.1016/s1470-2045(02)00679-4

[21] M. L. Kolhe, M. C. Trivedi, S. Tiwari, and V. K. Singh, Eds., Advances in Data and Information Sciences. Springer Singapore, 2018. [Online]. Available: https://doi.org/10.1007/978-981-10-8360-0

[22] C. Barata, M. E. Celebi, and J. Marques, "A survey of feature extraction in dermoscopy image analysis of skin cancer," IEEE Journal of Biomedical and Health Informatics, vol. 23, pp. 1096–1109, 2019.

[23] C. Barata, M. Celebi, and J. Marques, "Toward a Robust Analysis of Dermoscopy Images Acquired under Different Conditions," 2015, pp. 1–22.

[24] R. Kaur, "Real-time supervised detection of pink areas in dermoscopic images of melanoma: Importance of color shades, texture and location." pp. 466–473, 2015.

[25] A. Madooei, "Automatic detection of blue-white veil by discrete colour matching in dermoscopy images," pp. 453–460, 2013.

[26] K. Matsunaga, A. Hamada, A. Minagawa, and H. Koga, "Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble," CoRR, vol. abs/1703.0, 3 2017. [Online]. Available: https://arxiv.org/abs/1703.03108

[27] I. G. Díaz, "Incorporating the Knowledge of Dermatologists to Convolutional Neural Networks for the Diagnosis of Skin Lesions," CoRR, vol. abs/1703.0, 3 2017. [Online]. Available: http://arxiv.org/abs/1703.01976

[28] A. Menegola, J. Tavares, M. Fornaciali, L. T. Li, S. Avila, and E. Valle, "RECOD Titans at ISIC Challenge 2017," CoRR, vol. abs/1703.0, 3 2017. [Online]. Available: http://arxiv.org/abs/1703.04819

[29] A. Nozdryn-Plotnicki and J. Yap, "Ensembling Convolutional Neural Networks for Skin Cancer Classification," 2018.

[30] 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," CoRR, vol. abs/1808.0, 8 2018. [Online]. Available: http://arxiv.org/abs/1808.01694

[31] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[32] C. Barata and J. Marques, "Deep Learning for Skin Cancer Diagnosis with Hierarchical Architectures," IEEE International Symposium on Biomedical Imaging (ISBI), 2019.

[33] "ISIC 2018 Task 3: Training," 2019. [Online]. Available: https://challenge.kitware.com/#phase/%0A5abcbc6f56357d0139260e66

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

[35] T. Mitchell, Machine Learning. McGraw-Hill, Inc., 1997.

[36] Z. Zhang, P. Cui, and W. Zhu, "Deep Learning on Graphs: A Survey," {IEEE} Transactions on Knowledge and Data Engineering, p. 1, 2020. [Online]. Available: https://doi.org/10.1109/tkde.2020.2981333

[37] Y. Chen, Y. Lin, C. Kung, M. Chung, and I. Ten, "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes. Sensors (Basel)," Sensors, vol. 19, p. 2047, 2019.

[38] S. Khan, H. Rahmani, S. Shah, and M. Bennamoun, A guide to convolutional neural networks for computer vision. Morgan & Claypool Publishers, 2018. [Online]. Available: https://doi.org/10.2200/S00822ED1V01Y201712COV015

[39] A. L. Maas, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," 2013.

[40] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," CoRR, vol. abs/1702.0, 2 2017. [Online]. Available: http://arxiv.org/abs/1702.05659

[41] "Loss Functions — ML Glossary documentation," 2020. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[42] F. Chollet, Deep Learning with Python. Manning Publications, 2017.

[43] R. Babbar, I. Partalas, E. Gaussier, and M. R. Amini, "On Flat versus Hierarchical Classification in Large-Scale Taxonomies," in Advances in Neural Information Processing Systems 26, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1824–1832. [Online]. Available: http://papers.nips.cc/paper/5082-on-flat-versus-hierarchical-classification-in-large-scale-taxonomies.pdf

[44] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," {IEEE} Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, 2010. [Online]. Available: https://doi.org/10.1109/tkde.2009.191

[45] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics, 2010.

[46] “ISIC Challenge.” [Online]. Available: https://challenge.isic-archive.com/

[47] T. Hastie, The elements of statistical learning : data mining, inference, and prediction. New York: Springer, 2009.

[48] F. Chollet and others, “Keras,” 2015. [Online]. Available: https://keras.io

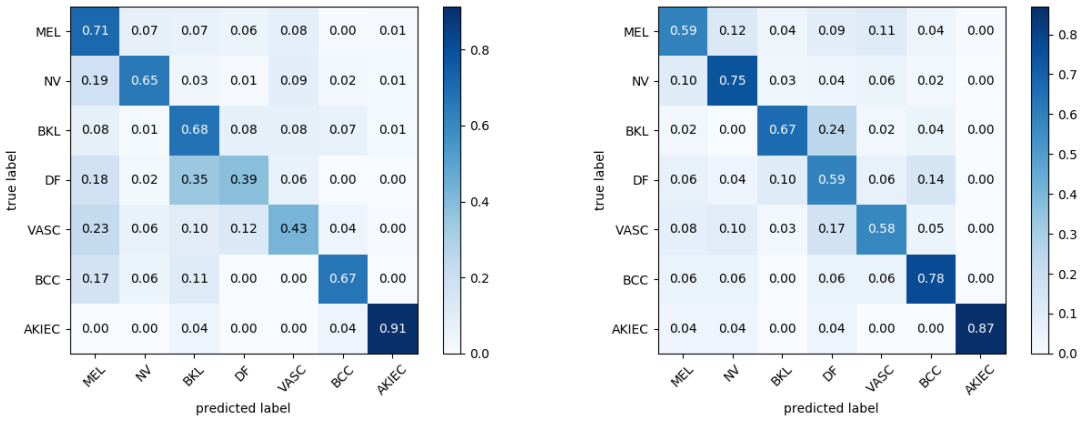
[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” 3 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

Appendix A - Flat Classifiers' Results

This appendix presents the results obtained in the validation set with flat classifiers implemented with VGG-19 and ResNet-101 from scratch and with transfer learning.

A.1 VGG-19

Figure A.1 presents the confusion matrices obtained with the VGG-19 model in the validation set, trained from scratch and with transfer learning, and table A.1 presents the metrics obtained.



(a) VGG-19 trained from scratch.

(b) VGG-19 trained with transfer learning.

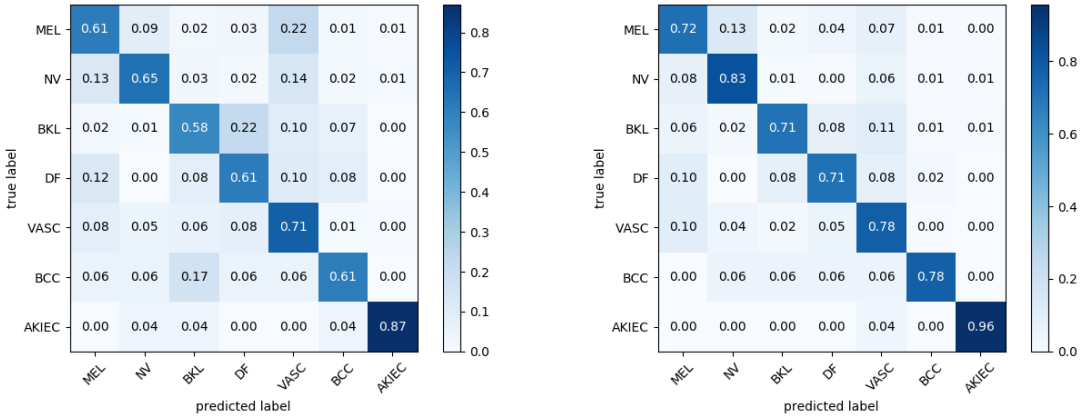
Figure A.1: Confusion matrices of VGG-19 obtained in validation set.

Table A.1: Metrics obtained with the VGG-19 model.

VGG-19	BACC	$P_{SUCC.}$
Transfer Learning	0.690	0.708
Scratch	0.634	0.633

A.2 ResNet-101

Figure A.2 presents the confusion matrices obtained with the ResNet-101 model in the validation set, trained from scratch and with transfer learning, and table A.2 presents the metrics obtained.



(a) ResNet-101 trained from scratch.

(b) ResNet-101 trained with transfer learning.

Figure A.2: Confusion matrices of ResNet-101 obtained in validation set.

Table A.2: Metrics obtained with the ResNet-101 model.

ResNet-101	**BACC**	$P_{SUCC.}$
Transfer Learning	0.786	0.809
Scratch	0.664	0.652

B

Appendix B - Project Code

This appendix presents some of the project code with the tensorflow framework [49]. First we create a flat model, and the hierarchical models with shared and non-shared features. Then, the flat model is compiled with the Adam optimizer and trained. Finally, the model loads the weights that led to the minimum validation loss during the training stage and evaluates it in the validation set.

Listing B.1: Project code.

```
import tensorflow as tf
from tensorflow.keras.applications.densenet import DenseNet121

# Create the flat model with DenseNet121, pre-trained in imagenet
densenet = DenseNet121(include_top=False, weights='imagenet',
    input_shape=(224, 224, 3))
model_flat = tf.keras.Sequential(densenet)
model_flat.add(tf.keras.layers.GlobalAveragePooling2D())
model_flat.add(tf.keras.layers.Dropout(0.5))
model_flat.add(tf.keras.layers.Dense(units=7, activation="softmax"))

# Create the hierarchical model with non-shared features
# Function to create each of the five classifiers
def create_classifier(no_classes):
    densenet = DenseNet121(include_top=False, weights='imagenet',
        input_shape=(224, 224, 3))
    model = tf.keras.Sequential(densenet)
    model.add(tf.keras.layers.GlobalAveragePooling2D())
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(units=no_classes, activation="softmax"))
    return model

# Create the five classifiers which are trained independently
# with different training sets
non_shared_a = create_classifier(2) # MEL, NMEL
non_shared_b = create_classifier(2) # NV, MELA
non_shared_c = create_classifier(2) # BEN, MAL
non_shared_d = create_classifier(3) # BKL, DF, VASC
non_shared_e = create_classifier(2) # AKIEC, BCC

# Create the hierarchical model with shared features
```

```
inputs = tf.keras.Input(shape=(224, 224, 3))
densenet = DenseNet121(include_top=False, weights='imagenet')
x = densenet(inputs)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.5)(x)
a = tf.keras.layers.Dense(2, activation='softmax', name="model_a")(x)
b = tf.keras.layers.Dense(2, activation='softmax', name="model_b")(x)
c = tf.keras.layers.Dense(2, activation='softmax', name="model_c")(x)
d = tf.keras.layers.Dense(3, activation='softmax', name="model_d")(x)
e = tf.keras.layers.Dense(2, activation='softmax', name="model_e")(x)
model_shared_features = tf.keras.Model(inputs=inputs,
    outputs=[a, b, c, d, e], name='shared')

# Compile the flat model with adam optimizer and learning rate = 1e-5
# using cross entropy loss
adam = tf.keras.optimizers.Adam(lr=1e-5)
model_flat.compile(loss=tf.keras.losses.categorical_crossentropy, optimizer=adam,
    metrics=['accuracy'])

# Create the scheduler and checkpoint callbacks
def scheduler(epoch):
    no_epochs = 20
    lr=1e-5
    if epoch < int(0.5*no_epochs):
        return lr
    elif epoch < int(0.75*no_epochs):
        return lr/10
    else:
        return lr/100

scheduler = tf.keras.callbacks.LearningRateScheduler(scheduler)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath='flat/cp.ckpt',
    save_weights_only=True, save_best_only=True)

# Initialization of dictionary that defines the class weights in the loss function
class_weight = {0: 24012./2697, 1: 24012./15936, 2: 24012./1272, 3: 24012./834,
    4: 24012./2625, 5: 24012./291, 6: 24012./357}

```

```
# Train the model with class_weights, for 20 epochs with a batch size = 10.
# The images and labels loading is omitted but they have the following types and shapes:
# x and x_val: array of train and validation images, respectively.
# Each image has shape = (224, 224, 3).
# y and y_val: list of train and validation labels, respectively.
# Each image is in the same index at x and y.
fit = model_flat.fit(x, y, batch_size=10, class_weight=class_weight,
    callbacks=[scheduler, checkpoint], epochs=20, shuffle=True,
    validation_data=(x_val, y_val))

# Load the model with the weights that led to the minimum loss in validation set
model_flat.load_weights('flat/cp.ckpt')

# Predict the categories of each example in validation set
y_pred = model_flat.predict_classes(x_val)

# Obtain and print the confusion matrix
conf_matrix = sklearn.metrics.confusion_matrix(y_val, y_pred)
print(conf_matrix)
```