André Georghton Cardoso Pacheco

Combining heterogeneous data and deep learning models for skin cancer detection

Vitória

2020

André Georghton Cardoso Pacheco

Combining heterogeneous data and deep learning models for skin cancer detection

Thesis presented to the Graduate Program in Computer Science of the Federal University of Espírito Santo as a partial requirement to obtain the degree of Doctor in Computer Science

Federal University of Espírito Santo - UFES

Graduate Program in Computer Science

Supervisor Prof. Dr. -Ing. Renato A. Krohling

Vitória

2020

Statement of Authorship

I hereby certify this thesis is based on my own work. All direct or indirect sources of information used are acknowledged as references, including graphs and datasets. Also, I have not submitted this thesis at any other institution in order to obtain a degree.

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Pacheco, Andre G. C., 1990-

P116c Combining heterogeneous data and deep learning models for skin cancer detection / Andre G. C. Pacheco. - 2020.

150 f. : il.

Orientador: Renato Antônio Krohling.

Tese (Doutorado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Deep Learning. 2. Data Aggregation. 3. Ensemble of Deep Models. 4. Convolutional Neural Networks. 5. Image Classification. 6. Skin Cancer Detection. I. Krohling, Renato Antônio. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Statement of Contributions for the creation of PAD-UFES-20 dataset

The skin lesion dataset PAD-UFES-20 created during this doctorate is a result of the effort of many people. First of all, I recognize the work of Professors Carlos Cley and Luiz F. S. de Barros who founded the Programa de Assistência Cirurgica e Dermatológica (PAD) in 1987. From this program all dataset were collected. Currently, Professor Patricia H. L. Frasson is the plastic surgeon and coordinator of the PAD. Professor Renato A. Krohling and Mr. Helder Knidel conceived the idea of this dataset and provided the computational infrastructure of the LABCIN. I led software technical development and, along with Gustavo R. Lima, Amanda S. Salomão, Igor P. Biral, Alana C. Simora, and Fábio C. R Alves Jr., coordinated the data collection. Breno A. Krohling, Gabriel G. de Angelo, José G. M. Esgario, Pedro B. C. Castro, and Felipe B. Rodrigues, worked on software development and data collection. Dr. Maria C. S. Santos coordinates histopathology examination and description. Dr. Rachel B. Espírito Santo, Dr. Telma L. S. Macedo, and Dr. Tânia R. P. Canuto are the dermatologists of the program who provided all clinical diagnostics for this dataset. I also recognize the effort of hundreds of medical students who work for the PAD by assisting in the appointments.

To my beloved grandmother wherever she is.

Acknowledgements

Foremost, I would like to profoundly thank my whole family, in particular, my wife Mariane, for the unconditional support, patience, and love; my parents, Almerinda and José, who did the impossible to provide me the best education a person can have; my parents-in-law, Jane e Marlucio, for all patience and understanding; the support of my brother Lucas; my sisters-in-law Natalia and Marilia; sister-in-law's husband Willyam, all my cousins, uncles, and aunts; and my grandmother Nadir who has recently left us, but I am sure she would be proud of me.

I would like to express my deep gratitude to my supervisor Prof. Renato A. Krohling for the stimulus and guidance that made this work possible. To all doctors and students from the Programa de Assistência Cirúrgica e Dermatológica (PAD-UFES), in particular Dr. Patrícia, Dr. Rachel, Dr. Luiz, and the (former) monitors Thales, Marcela, Elionay, Fabio, Igor, Jayme, Gustavo, and Amanda. I thank professors Thomas Trappenberg and Saagev Oore for having me during my visiting research at Dalhousie University. Professors Celso A. S. Santos and Daniel C. Cavalieri, who provided valuable observations to improve this work, and Professors João Paulo Papa, Vinícius F. S. Mota, Maria C. S. Boeres, and Mariella Berger for accepting the invitation to be part of the examination board. And the foment research agencies CAPES and FAPES, which provided the financial support to my scholarship and my doctoral project.

Finally, my sincere thanks to my lab partners Gabriel, Guilherme(s), Giuliano, Breno, Carlos, Lucas, Pedro, Leandro, Arnauld, Yoshi, Zaher, and Chandramouli for the technical support and advice. To Ivan and Rodolfo for all discussions about science, research, and academic life. To the last but not least, my great friends Julio, Glicia, Félix, Luis, Mateus, Jordana, Daniel, Aluisio, Gustavo, Pedro, Ramon, Dudu, and all my friends from EngComp-09. All of you play an important role in my life and are a big part of this work.

"We strive toward knowledge, always more knowledge, but must understand that we are, and will remain, surrounded by mystery."

Marcelo Gleiser

Abstract

Currently, Deep Neural Networks (DNN) are the most successful and common methodologies to tackle medical image analysis. Despite the success, applying Deep Learning for these types of problems involves several challenges such as the lack of large training datasets, data variance, and noise sensitivity. In this thesis, our main focus is on proposing solutions to assist Deep Learning models to deal with these issues when they are applied to medical (clinical) image problems, in particular for skin cancer detection. Basically, we work on two main topics: data classification using images and context meta-data and dynamic weighting for an ensemble of deep models. First, we propose two methods to combine images and meta-data; one method is based on features concatenation that uses a mechanism to balance the contribution of each source of data; the second method, named Meta-data Processing Block (MetaBlock), uses meta-data to support the classification by identifying the most relevant features extracted from the images. Next, we propose an approach, based on a Dirichlet distribution and Mahalanobis distance, to learn dynamic weights for an ensemble of deep models. The learned weights are used to reduce the impact of weak models on the aggregation operator and to online select models from the ensemble. All these methods are evaluated in well-known image classification datasets in different experiments. Results show that the proposed methods are competitive with other approaches that deal with the same problems. Lastly, we carry out a case study using a new skin lesion dataset – composed of clinical images collected from smartphones and patient demographics – collected in partnership with the Dermatological and Surgical Assistance Program of the Federal University of Espírito Santo. Results achieved using this dataset are comparable to other recent performance reported in the literature, which shows that the proposed algorithms are viable to deal with skin cancer detection.

Keywords: Deep Learning, Data Aggregation, Ensemble of Deep Models, Convolutional Neural Networks, Image Classification, Skin Cancer Detection.

Resumo

Atualmente, as Redes Neurais Profundas (RNP) são os modelos que apresentam os melhores resultados para lidar com a análise de imagens médicas. Apesar do sucesso, a aplicação de Aprendizado Profundo para esses tipos de problemas apresenta vários desafios, como a falta de grandes conjuntos de dados de treinamento, variação de dados e sensibilidade ao ruído. O foco principal deste trabalho é propor soluções para auxiliar os modelos de Aprendizado Profundo a lidar com esses desafios quando aplicados a problemas que lidam com imagens (clínicas) médicas, em particular a detecção de câncer de pele. De maneira geral, as propostas são feitas em dois tópicos principais: classificação de dados utilizando imagens e metadados do contexto e ponderação dinâmica para um conjunto de modelos profundos. Primeiro, foi proposto dois métodos para combinar imagens e metadados; um método é baseado na concatenação de atributos que utiliza um mecanismo para equilibrar a contribuição de cada fonte de dados; o segundo método, denominado Bloco de Processamento de Metadados (MetaBlock), utiliza os metadados para apoiar a classificação, identificando os atributos mais importantes extraídos das imagens. Em seguida, propomos uma abordagem, baseada na distribuição de Dirichlet e na distância de Mahalanobis, para aprender dinamicamente os pesos para um conjunto de modelos profundos. Esses pesos são utilizados para reduzir o impacto de modelos ruins no operador de agregação e para selecionar modelos do conjunto de maneira online. Todos os métodos propostos são avaliados em diferentes bases de dados de classificação considerando diferentes experimentos. Os resultados obtidos mostram que os métodos propostos são competitivos com outras abordagens que lidam com os mesmos problemas. Por fim, é realizado um estudo de caso utilizando uma nova base de dados de lesões de pele - composta por imagens clínicas coletadas via smartphones e informações clínicas dos pacientes - coletados em parceria com o Programa de Assistência Dermatológica e Cirúrgica (PAD) da Universidade Federal do Espírito Santo (UFES). O desempenho obtido para essa base de dados é comparável com outros resultados recentemente reportados na literatura, o que indica que os algoritmos propostos são viáveis para lidar com detecção de câncer de pele.

Keywords: Aprendizado profundo, Agregação de Dados, Conjunto de Modelos Profundos, Redes Neurais Convolutivas, Classificação de imagens, Detecção de Câncer de Pele.

List of Figures

List of Tables

Contents

1 Introduction

Long before the invention of automated machines, ancient civilizations have imagined robots and artificial lives that were *made, not born* (Mayor, 2018). Since the first programmable computer was created, the idea of intelligent machines has risen (Lovelace, 1842). Turing (1950) proposed an *imitation game* – also known as Turing test – to address the philosophical question: *can machines think?* In his work, Turing introduced key concepts that opened the doors to a new field that later would be named Artificial Intelligence (AI), which may be defined as *the effort to automate intellectual tasks normally performed by humans* (Chollet, 2018). Pioneers works in AI (Rosenblatt, 1958; Rosenblatt, 1961; Minsky, 1961; Ivakhnenko; Lapa, 1966; Minsky; Papert, 1969) successfully solved problems that are difficult for human beings but relatively straightforward for computers (Goodfellow; Bengio; Courville, 2016) – for example, a large sequence of hardcode rules to play chess. Later, it turned out that the real challenge for AI algorithms is to solve problems that are easy for human beings, such as recognizing traffic signs on the streets. In order to deal with these kinds of problems, a new paradigm called Machine Learning (ML) has risen. ML models learn hierarchy representations and concepts directly from data – without specific rules to solve a given problem. In some cases, successive layers of representations are built on top of each other, which allows the model to learn complex concepts from simpler ones (Goodfellow; Bengio; Courville, 2016; Chollet, 2018; Sejnowski, 2018). As the number of layers increases, the model becomes deep, which defines an ML subfield called Deep Learning (DL).

Although Deep Learning algorithms date back to the 1980s (Fukushima, 1980; Schmidhuber, 2015), they only became truly viable over the last decade, mainly because of the increase of computational power and availability of large datasets (Faes et al., 2019). Since 2012, different Deep Learning models have been presenting important advances for several tasks, such as computer vision (Krizhevsky; Sutskever; Hinton, 2012; Long; Shelhamer; Darrell, 2015; Wu et al., 2019a), speech recognition (Chiu et al., 2018; Nassif et al., 2019; Wu; Sahoo; Hoi, 2020), medical image analysis (Ronneberger; Fischer; Brox, 2015; Esteva et al., 2017; Ardila et al., 2019), natural language processing (NLP) (Radford et al., 2019; Devlin et al., 2018), digital games (Berner et al., 2019; Vinyals et al., 2019), among many others (Dargan et al., 2019). Many of these models present results comparable to or better than human expert performance. Standard Deep Learning models include the Restricted Boltzmann Machine (RBM) (Hinton, 2002; Hinton; Salakhutdinov, 2006), Convolutional Neural Networks (CNN) (LeCun; Bengio et al., 1995; LeCun et al., 1998), Autoencoder (Rumelhart; Hinton; Williams, 1985; Bengio; LeCun et al., 2007), Generative Adversarial Networks (GAN) (Schmidhuber, 1992; Goodfellow et al., 2014), Long Short-

Term Memory (LSTM) (Hochreiter; Schmidhuber, 1997), and Capsule Networks (Sabour; Frosst; Hinton, 2017).

Nowadays, Deep Neural Networks (DNN) are the most successful methodologies to tackle medical image analysis (Litjens et al., 2017). State-of-the-art results have been reported for chest radiography diagnosis (Irvin et al., 2019), lung cancer prediction (Ardila et al., 2019), diabetic retinopathy grading (Sahlsten et al., 2019), breast cancer detection (Shen et al., 2019), Alzheimer's disease classification (Oh et al., 2019), skin cancer detection (Tschandl et al., 2019; Pacheco; Krohling, 2020a), among others. Nonetheless, applying Deep Learning for these types of problems presents several challenges such as the lack of large training datasets, data variance, and noise sensitivity. In this thesis, our main focus is on proposing solutions to assist Deep Learning models to deal with these issues when they are applied to medical image analysis, in particular for skin cancer detection.

1.1 Skin cancer detection

Skin cancer occurs when skin cells are damaged, for example, by overexposure to ultraviolet (UV) radiation from the sun (Bray et al., 2018). Unlike other forms of cancer, the reporting of skin cancer diagnoses is not required by most cancer registries worldwide (Siegel; Miller; Jemal, 2019). Regardless, the World Health Organization (WHO) estimates that one in every three instances where a person is diagnosed with cancer is a skin cancer (WHO, 2019). Over the recent decades, in countries such as Canada, USA, and Australia, the number of people diagnosed with skin cancer has been increasing at a constant rate (CCS, 2014; CCA, 2018; ACS, 2019). In Brazil, according to the Brazilian Cancer Institute (INCA), skin cancer is the most common dysplasia and for 2020-2022 it is expected 177 thousand new cases across the nation (INCA, 2020). In Figure 1 is depicted the estimated age-standardized incidence of skin cancer in the world.

There are three main types of skin cancer: Basal cell carcinoma (BCC), squamous cell carcinoma (SCC), and melanoma (MEL). Melanoma is the rarest one; however, because of the high level of metastasis[1], it is the most lethal one. On the other hand, BCC and SCC, which are known as non-melanoma skin cancers (NMSC), represent the major skin cancer occurrence worldwide. As they rarely metastasize, they have low lethality risk (CCA, 2018). Despite the lethality difference between both categories, the number of deaths caused by both groups is similar. For instance, in 2017 the Brazilian government registered 2,250 deaths caused by melanoma against 1,835 by non-melanomas (INCA, 2020). Even with low lethality risks, the higher NMSC's incidence justifies a similar number of deaths.

In order to diagnose skin cancer, dermatologists screen the suspicious lesion on the skin using their experience to identify it. Early detection is fundamental to increase patient

[1] when damaged cells invade other parts of the body via blood vessels and lymph vessels

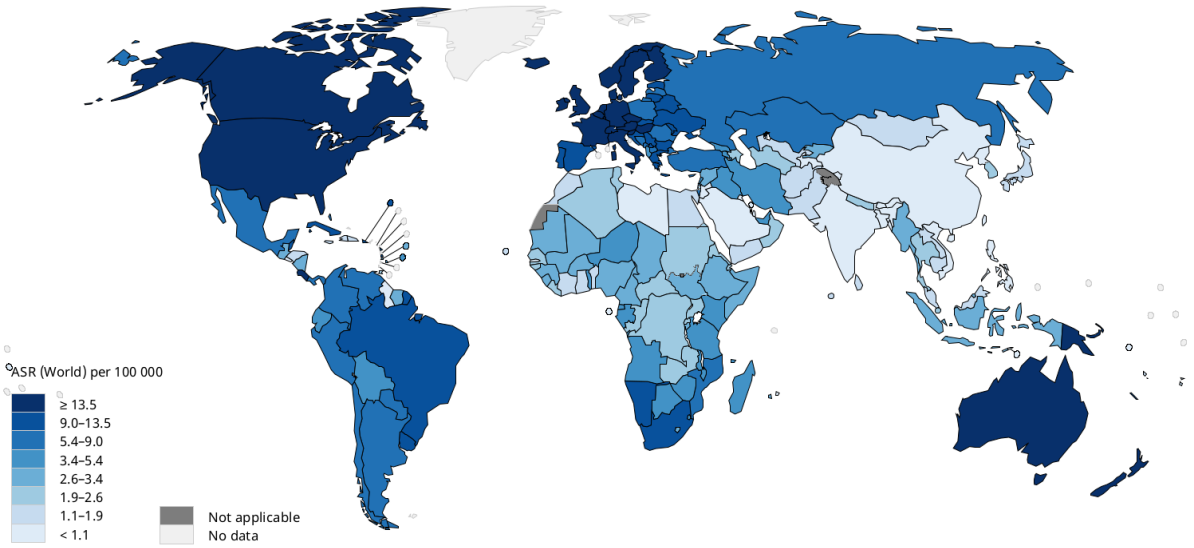


Figure 1 – Estimated age-standardized incidence of skin cancer in the world for both sexes and ages 0-74 (Bray et al., 2018). Countries in Europe, USA, Canada, New Zealand, and Australia have the highest rates. In South America, Brazil and Uruguay present the highest estimate occurrences.

prognostics, in particular for melanoma. Accurate diagnosis is challenging and requires proper training and experience in dermoscopy – a non-invasive diagnostic technique that uses a medical device named dermatoscope to magnify subsurface structures of the skin revealing morphologic features that are normally not visible to the naked eye (Argenziano; Soyer, 2001). In Figure 2 is illustrated an example of clinical and a dermoscopy image. Kittler et al. (2002) and Sinz et al. (2017) showed that dermoscopy substantially increases diagnostic accuracy. However, it depends on the dermatologist's experience level, i.e., less experienced examiners do not present improvement using dermoscopy. As such, the high incidence rate of skin cancer and the lack of experts and medical devices, particularly in rural areas (Feng et al., 2018) and emerging countries (Scheffler et al., 2008), have increased the demand for computer-aided diagnosis (CAD) systems for skin cancer detection.

Over the past decades, computer-aided diagnosis (CAD) systems for skin lesion analysis have been intensively investigated and have demonstrated to be a promising tool. Pioneer works, such as (Umbaugh et al., 1993), (Ercal et al., 1994) and (Green et al., 1994), used low-level handcrafted features to distinguish melanomas and non-melanoma cancers. Later, several computational approaches have been developed based on ABCD(E) rule, pattern analysis, and 7-point checklist, which are common methods used by dermatologists to diagnose skin cancer (Argenziano et al., 1998; Masood; Al-Jumaily, 2013). These approaches are mostly based on traditional computer vision algorithms to extract various features such as shape, color, and texture (Celebi et al., 2007; Wighton et al., 2011; Maglogiannis; Delibasis, 2015; Barata; Celebi; Marques, 2014; Oliveira et al., 2018), to feed a classifier, for example, a Support Vector Machine (SVM) (Scharcanski;

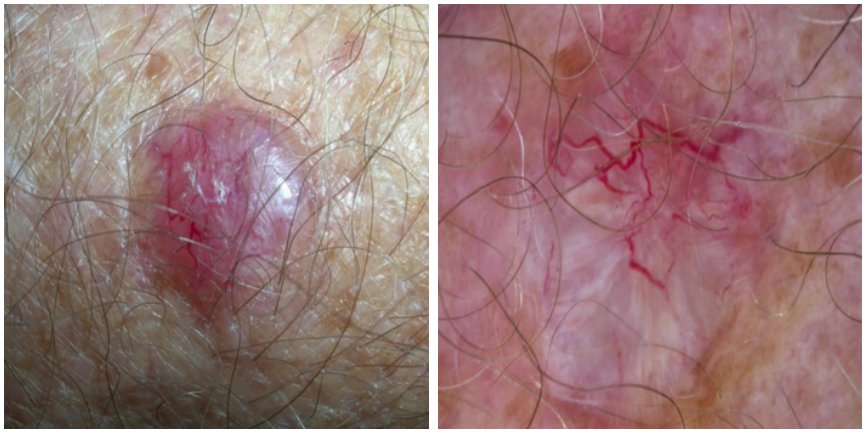


Figure 2 – The differences between a clinical image (left) and a dermocopy image (right) of the same skin lesion. As we can see, the dermoscopy one present more details than the clinical one. Source: (Marghoob; Usatine; Jaimes, 2020).

Celebi, 2013; Codella et al., 2015; Spyridonos et al., 2017). Two limitations may be pointed out in these approaches. First, the ABCD(E) rule and the 7-point checklist are designed only for pigmented lesions, which means they cannot be used to diagnose BCC or SCC, for example. Second, the handcrafted features extracted by these methods have limited generalization capability (Yu et al., 2017).

As previously stated, Deep Learning models have recently become viable for medical image analysis. As a consequence, several models have been proposed to deal with skin cancer detection. Yu et al. (2017) presented a very deep CNN and a set of schemes to learn under limited training data. Esteva et al. (2017) used a pre-trained GoogleNet CNN architecture (Szegedy et al., 2016) to train over 120 thousand images and achieved a dermatologist-level diagnostic. Haenssle et al. (2018) and Brinker et al. (2019) also used pre-trained models to compare their performance to dermatologists. In both works, the models have shown competitive or outperformed the dermatologists performance. Other efforts using deep learning have been made to detect skin cancer, such as an ensemble of models (Codella et al., 2017b; Harangi, 2018; Pacheco; Krohling, 2020b), feature aggregation of different models (Yu et al., 2019), among others (Attia et al., 2017; Han et al., 2018; Nida et al., 2019; Angelo; Pacheco; Krohling, 2019; Serte; Demirel, 2019). The majority of these works are based on dermoscopy images, mainly for three reasons:

1. There is an open well-known dataset, provided by the International Skin Imaging Collaboration (ISIC) (Codella et al., 2017a; Tschandl; Rosendahl; Kittler, 2018; Combalia et al., 2019), containing over 30 thousand skin lesion images.

2. Collect a dataset composed of clinical skin lesion images – those ones taken from standard cameras – is a quite hard task since it demands time to grow the dataset and multidisciplinary effort.

3. It is easier to work with dermoscopy images since they provide more details about the skin surface and they are not affected by illumination or camera resolution – see Figure 2.

1.2 Motivation

Developing CAD systems to work with dermoscopy images is important; however, in most remote places – such as emerging countries and rural areas – there is a lack of experts and dermatoscope available to clinicians. As a result, these systems are not applicable to those places. In this context, smartphones may be a low-cost solution to assist non-expert clinicians to provide the first screening for people who live in these remote places. Nonetheless, to embed a CAD system in a smartphone it must work with clinical images instead of dermoscopy ones. This involves, first, the creation of a clinical image dataset and the design of new methods to assess this type of data.

An important clue towards a more accurate clinical diagnosis is the patient demographics. In fact, dermatologists do not trust only on image screening; they also take into account clinical information such as patient's age, anatomical region of the skin lesion, if the lesion bleeds, among many others (Wolff et al., 2017; Azulay, 2017). Brinker et al. (2018) presented a review for deep learning models applied to skin cancer detection and concluded that an improvement in classification quality could be achieved by adding patient demographics into the classification process. Nonetheless, most of the reported works consider only the skin image to yield the prediction.

Kharazmi et al. (2018), Liu et al. (2019), and Pacheco and Krohling (2020a) proposed models to combine both skin lesion images and patient demographics. Although they report promising improvements, all of these works combine both data using feature concatenation, which may not take into account the potential relationship between meta-data and the visual features extracted from the images. Recently, Li et al. (2020) proposed the MetaNet: a multiplication-based fusion approach that uses 1D convolution on meta-data to combine them with the image features. The authors reported promised results for skin cancer detection, but only for some cases of this disease. In short, there is still room for improvement, and better aggregation methodologies must be explored.

1.3 Objectives and hypothesis

The main objective of this thesis is to develop algorithms to improve medical image classification using Deep Learning models and apply them to skin cancer detection. In particular, we propose solutions to deal with the following two problems:

- **Combining images and meta-data:** some image classification problems are based

on images and meta-data – for example, in skin cancer detection, we may have images from the skin lesion and patient demographics to support the detection. In this sense, it is necessary to develop an approach to consider both types of data in the classification process.

- **Dynamic ensemble weighting:** a common strategy to improve the classification performance and robustness is to trust in predictions from an ensemble of different deep models. The most common ensemble aggregation operators assign the same importance to all models in the ensemble. As a consequence, they cannot identify weak models that may negatively influence the ensemble performance. Dynamic ensemble weighting is important to identify and reduce the influence of these weak models on the ensemble output.

We propose solutions to these problems to answer two main hypothesis in this thesis. First, is it possible to overcome the lack of details in clinical skin cancer images with meta-data and achieve a classification performance as good as the one obtained for dermoscopy images? Second, can we improve the image classification provided by an ensemble of deep models by determining the relevance of each classifier on the fly? These hypothesis are addressed and discussed in Chapters 3 and 4, respectively.

1.4 Contributions

The main contributions of this thesis are summarized as follows:

- We propose two methods to deal with data classification based on images and meta-data.
  - The first method is based on features concatenation and uses a mechanism to control the contribution of each source of data before classification.
  - The second method, named Meta-data Processing Block (MetaBlock), uses meta-data to support the classification by identifying the most relevant features extracted from the images without using concatenation.
  - We show that meta-data, when available, is quite important to improve the performance of skin cancer predictors based on deep learning models.
- We develop an approach, based on a Dirichlet distribution and Mahalanobis distance, to learn dynamic weights for an ensemble of deep models
  - The learned weights are used to reduce the impact of weak models on the aggregation operator.

  - It is also possible to apply online ensemble selection/pruning.

- In partnership with the Dermatological and Surgical Assistance Program (in Portuguese: *Programa de Assistência Cirúrgica e Dermatológica* - PAD) of the Federal University of Espírito Santo (in Portuguese: Universidade Federal do Espírito Santo - UFES), we developed a web-based system and a multi-platform smartphone application to collect skin lesion images and patient demographics.

  - From this software, we created a new dataset composed of clinical images and patient demographics. This dataset is used as a case study in this thesis and is made publicly available to support future research. As far as we know, this is the first public dataset containing both types of data.
  - Beyond data collection, the software allows clinicians to track patient lesions and obtain statistics to improve their understanding of skin diseases. Previously, most of these data were lost or collected and store in an unstructured way.

1.5 List of publications

In this section is presented the list of publications achieved during the doctorate.

1. **Aggregation of neural classifiers using Choquet integral with respect to a fuzzy measure**. Published in Neurocomputing (Pacheco; Krohling, 2018a).

2. **Restricted Boltzmann machine to determine the input weights for extreme learning machines**. Published in Expert Systems with Applications (Pacheco; Krohling; Silva, 2018).

3. **An approach to improve online sequential extreme learning machines using restricted Boltzmann machines**. Published in IEEE International Joint Conference on Neural Networks (IJCNN) (Pacheco; Krohling, 2018b).

4. **Skin lesion segmentation using deep learning for images acquired from smartphones**. Published in IEEE International Joint Conference on Neural Networks (IJCNN) (Angelo; Pacheco; Krohling, 2019).

5. **Skin cancer detection based on deep learning and entropy to detect outlier samples**. Published in Medical Image Computing and Computer Assisted Intervention (MICCAI) at Skin Lesion Analysis Towards Melanoma Detection challenge (Pacheco; Ali; Trappenberg, 2019) – 4th and 3rd place in ISIC challenge tasks 1 and 2, respectively.

6. **Recent advances in deep learning applied to skin cancer detection**. Published in Neural Information Processing Systems (NeurIPS) at Retrospectives workshop (Pacheco; Krohling, 2019).

7. **The impact of patient clinical information on automated skin cancer detection**. Published in Computers in Biology and Medicine (Pacheco; Krohling, 2020a).

8. **Learning dynamic weights for an ensemble of deep models applied to medical imaging classification**. Published in IEEE International Joint Conference on Neural Networks (IJCNN) (Pacheco; Krohling, 2020b).

9. **On Out-of-Distribution Detection Algorithms with Deep Neural Skin Cancer Classifiers**. Published in IEEE Computer Vision and Pattern Recognition (CVPR) at Skin Image Analysis Workshop (ISIC) workshop (Pacheco et al., 2020b) – **Best Paper Award.**

10. **An App to Detect Melanoma Using Deep Learning: An Approach to Handle Imbalanced Data Based on Evolutionary Algorithms**. Published in IEEE International Joint Conference on Neural Networks (IJCNN) (Castro et al., 2020).

11. **PAD-UFES-20: a skin lesion benchmark composed of patient data and clinical images collected from smartphones**. Published in Data in Brief (Pacheco et al., 2020a).

12. **An attention-based mechanism to combine images and metadata in deep learning models applied to skin cancer classification**. Published in IEEE Journal of Biomedical and Health Informatics (Pacheco; Krohling, 2021).

1.6 Software implementations

We used different open source frameworks to implement the algorithms proposed/used in this work. All these frameworks use Python as the base programming language. To handle general data, we use NumPy and Pandas. To deal with images, we use OpenCV and Pillow. For general machine learning metrics and algorithms, we perform the scikit-learn package. All visualizations are achieved using Matplotlib and Seaborn. Lastly, everything that involves deep learning models is implemented by using Pytorch. All code we developed is available on Github[2].

In terms of hardware, the experiments performed in Chapter 4 used a NVIDIA Tesla V100 16GB from the Compute Canada servers infrastructure. The remaining experiments

[2] Avalaible on <https://github.com/paaatcha/raug> and on <https://github.com/paaatcha/my-thesis>

were performed using a NVIDIA GTX 1060 6GB and a NVIDIA Titan X 12GB from the Laboratório de Computação e Engenharia Inspirado na Natureza (LABCIN-UFES) infrastructure.

1.7 Thesis organization

The remaining of this thesis is organized as follows: in Chapter 2, we introduce fundamentals concepts in Deep Learning and describe the Convolutional Neural Network (CNN), the deep model explored in this work. The proposed algorithms are described in the next two chapters: in Chapter 3, we describe the algorithms to combine images and meta-data for data classification; and in Chapter 4 we describe the algorithm to learn dynamic weights for an ensemble of deep models. Each of these chapters has an experiment section in which we evaluate each of the proposed approaches. Next, in Chapter 5, we carry out a case study for skin cancer detection using the dataset we created during this doctorate. This dataset is composed of clinical images and patient demographics – all software we developed to collect and store the data is described in Appendix A. Finally, in Chapter 6 we draw our conclusions and indicate directions that should be addressed in the future.

2 Fundamental concepts on Machine and Deep Learning

In this chapter, we introduce the fundamental concepts regarding Machine and Deep learning. We start by describing the types of learning, learning theory, and Maximum Likelihood Estimation (MLE). In the following, we introduce Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN).

2.1 Fundamentals of Deep Learning

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are terms in which the concepts overlap each other. In brief, we may define each field as follows (Jeffcock, 2018):

- **Artificial Intelligence (AI):** it consists of algorithms that allow computers to mimic human behavior.
- **Machine Learning (ML):** this is a way to achieve AI, but learning concepts directly from data, i.e., with minimal human intervention.
- **Deep Learning (DL):** this is an ML's subfield in which models are composed of stacks of artificial neural layers that are usually trained using an end-to-end optimization algorithm. The term deep refers to the high number of layers.

The paradigm difference between ML and AI is illustrated in Figure 3.

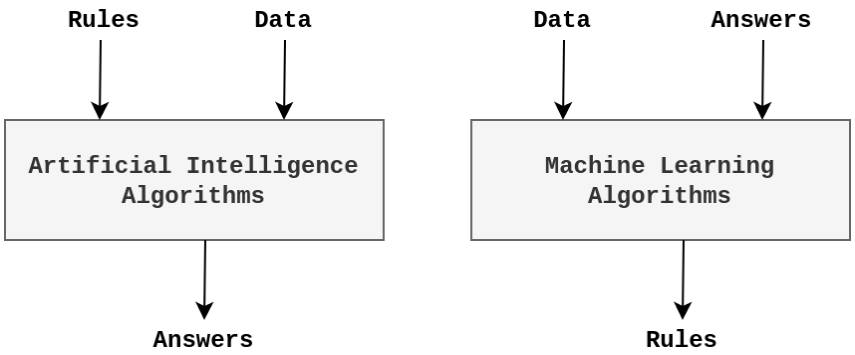


Figure 3 – The paradigm difference between ML and AI. While AI is based on hardcoded rules to output answers, ML learns these rules from the data and the answers. This figure is an adaptation from (Chollet, 2018)

As Deep Learning is a subfield of Machine Learning, they share many basic principles, which we will discuss in this section. Mitchell (1997) defined Machine Learning as follows:

> *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.*

The task T is the problem that an algorithm must learn to solve. Learning means identify a function $f : X \to Y$ that maps the input distribution X to the output distribution Y. Tasks in ML are usually described in terms of how an algorithm should process a new sample $x \in X$ (Goodfellow; Bengio; Courville, 2016). Examples of tasks that Machine Learning can solve include clusterization, regression, denoising, density estimation, and classification, which is the main focus of this thesis.

The performance measure P is the metric in which the algorithms' effectiveness is evaluated. Obviously, it depends on task T. For example, in the classification task, a common performance measure is the accuracy, which is the proportion of samples $x \in X$ that the algorithm predicts the correct output $y \in Y$.

The experience E is the set of data containing examples that are used to train (the process of learning concepts from data), to validate, and to test a Machine Learning algorithm.

2.1.1 Types of learning

Machine Learning algorithms can be clustered into three main types of learning: supervised learning, unsupervised learning, and reinforcement learning (Goodfellow; Bengio; Courville, 2016; Géron, 2019):

- **Supervised learning**: algorithms are trained on datasets that include input samples and target solutions – also known as labels. Classifying skin diseases is an example of this type of learning.
- **Unsupervised learning**: algorithms are trained on datasets that include input samples without target solutions. Clustering objects by color is an example of this type of learning.
- **Reinforcement learning**: algorithms are trained by observing the environment, selecting and performing actions, and getting rewards or penalties. Basically, the algorithm must learn a strategy to solve a problem by itself. Algorithms that learns how to play a computer game by itself is an example of this type of learning.

In addition to these three main types of learning, it is worth pointing out two other approaches: semi-supervised and self-supervised learning. Semi-supervised learning is a combination of supervised and unsupervised ones. In brief, semi-supervised algorithms are trained using both labeled and unlabeled samples. On the other hand, self-supervised algorithms aim to learn pattern features using automatically generated labels without any human annotations. Essentially, it may be considered as a sub-field of unsupervised learning in which a machine learning model can be trained by learning objective functions of pretext tasks and the features are learned through this process (Jing; Tian, 2020).

2.1.2 Data classification

Classification consists in identifying one or more labels to a new sample. Every sample is composed of a set of features $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ and one or more labels $Y = \{y_1, \cdots, y_N\}$, where N is the number of samples. As previously stated, in Machine Learning, classification algorithms are trained in a supervised way. In this sense, the algorithms' ultimate goal is to learn a function $f(\mathbf{x_i}; \boldsymbol{\theta})$ that predicts the label y_i to the sample $\mathbf{x}_i$, with $i = \{1, \cdots, N\}$. In other words, the algorithm aims to learn a distribution $p(y_i \mid \mathbf{x}_i; \boldsymbol{\theta})$ in which the probabilities can be interpreted as the likelihood or confidence of $\mathbf{x}_i$ belonging to the label y_i and $\boldsymbol{\theta}$ is the distribution parameters that control the algorithm behavior.

A particular instance of this task is image classification, which is obviously the task of assigning a label to an image. This is one of the most important problem in computer vision and has several practical applications (Li et al., 2019a). Recognizing visual concepts on images is such an easy task for human beings. Nonetheless, from the perspective of a computer vision algorithm, there are several challenges involving this task, for example, viewpoint variation, occlusion, illumination, scale variation, among others (Li et al., 2019a). The most successful Machine Learning/Deep Learning approach to deal with this task is the Convolutional Neural Network (CNN), which we describe in the next sections.

2.1.3 Generalization

In order to train Machine Learning algorithms, it is common to split the set of data available to fit the model into training, validation, and test partitions. The training partition is the set of data that algorithms learn from; the validation is a small part of the training partition that is used to certify if the algorithm is really learning; lastly, the test partition simulates the real world. The ability of ML models to correctly recognize test samples is known as generalization. This concept is fundamental to ensure that trained models work properly in the real world.

Given some error measure – a metric that computes the difference between predicted

and real outputs – we can compute the training, validation, and test error. It is crucial to reduce not only the training error but also the validation and test ones. The expected value of test error is known as generalization error. Successful ML models must present low test errors. This is what distinguishes Machine Learning from the optimization field (Goodfellow; Bengio; Courville, 2016).

In Machine Learning, one usually assumes that the train and test sets are *independently and identically distributed* (iid) (Smola; Vishwanathan, 2008; Bengio et al., 2009). It means both sets are drawn from the same probability distribution. This is important in statistical learning theory since it allows sampling from the training partition to fit $f(\mathbf{x}_i; \boldsymbol{\theta})$ aiming to reduce the generalization error. In this context, the expected test error is always greater than or equal to the expected value of training error. Therefore, it is important to ML algorithms ensure both a low training error and a low difference between training and test errors (Smola; Vishwanathan, 2008; James et al., 2013; Goodfellow; Bengio; Courville, 2016). This concept leads us to two other important definitions: 1) the trade-off between bias and variance; 2) underfitting and overfitting.

2.1.3.1 Bias and variance

In Statistical and Machine Learning it is important to understand the trade-off between bias and variance. Bias may be represented by the training error and the variance by the test error. A model with high bias does not fit well the training data and, consequently, the estimated distribution is far from the real one. On the other hand, a model with high variance pays too much attention to training data and does not generalize well, i.e., it has a high test error.

Let us consider one aims to fit a model to predict an output Y from an input X. The relationship between Y and X is described by a function $g(X)$. Thus, we may assume:

$$Y = g(X) + \epsilon \tag{2.1}$$

where ϵ is an irreducible error, i.e., a part of Y that is not predictable from X regardless how well $g(X)$ is estimated – because of noise, for example. An estimation function for $g(X)$ is represented by $f(X)$ and an estimation/prediction of Y is described as:

$$\hat{Y} = f(X) \tag{2.2}$$

We may compute the expected squared error of the estimated model $f(X)$ as follows

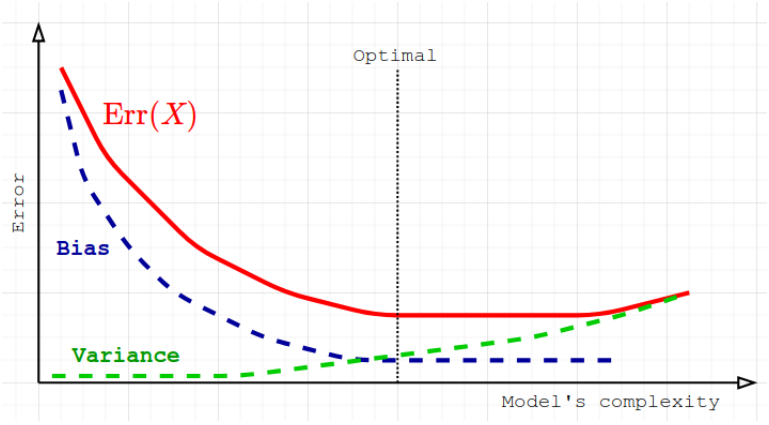


Figure 4 – Illustration of the bias-variance trade-off. As the model's complexity increases, the bias tends to decrease and variance to increase. Conversely, as it decreases, bias tends to increase and variance to decrease. The optimal point is a balance of all errors. Adapted from (Goodfellow; Bengio; Courville, 2016).

(James et al., 2013):

$$
\begin{aligned}
\mathbb{E}\left[Err(X)\right] &= \mathbb{E}\left[\left(Y-\hat{Y}\right)^2\right] \\
\mathbb{E}\left[Err(X)\right] &= \mathbb{E}\left[\left(Y-\mathbb{E}\left[\hat{Y}\right]+\mathbb{E}\left[\hat{Y}\right]-\hat{Y}\right)^2\right] \\
\mathbb{E}\left[Err(X)\right] &= \mathbb{E}\left[\left(Y-\mathbb{E}\left[\hat{Y}\right]\right)^2+\left(\mathbb{E}\left[\hat{Y}\right]-\hat{Y}\right)^2+2\left(Y-\mathbb{E}\left[\hat{Y}\right]\right)\left(\mathbb{E}\left[\hat{Y}\right]-\hat{Y}\right)\right] \\
\mathbb{E}\left[Err(X)\right] &= \mathbb{E}\left[\left(Y-\mathbb{E}\left[\hat{Y}\right]\right)^2+\left(\mathbb{E}\left[\hat{Y}\right]-\hat{Y}\right)^2\right]+\mathbb{E}\left[2\left(Y-\mathbb{E}\left[\hat{Y}\right]\right)\left(\mathbb{E}\left[\hat{Y}\right]-\hat{Y}\right)\right]^{0} \\
\mathbb{E}\left[Err(X)\right] &= \left(\underbrace{Y-\mathbb{E}\left[\hat{Y}\right]}_{\text{Bias}}\right)^2+\underbrace{\mathbb{E}\left[\left(\mathbb{E}\left[\hat{Y}\right]-\hat{Y}\right)^2\right]}_{\text{variance}}
\end{aligned}
\tag{2.3}
$$

From Equation 2.3, we observe that to minimize $Err(X)$ it is necessary to estimate a function $f(X)$ that simultaneously produces low bias and variance. Nonetheless, these errors are connected to the model's complexity (also known as model's capacity), as it increases, the bias tends to decrease and variance to increase. On the other hand, as it decreases, bias tends to increase and variance to decrease. This relationship, illustrated in Figure 4, is known as the bias-variance trade-off and it is one of the most important concepts in Machine Learning (Goodfellow; Bengio; Courville, 2016).

2.1.3.2 Underfitting and overfitting

Underfitting occurs when a model is quite simple to properly learn patterns on data. It presents poor performance both on training and test partitions. Usually, it has a high bias and variance. Potential solutions to this problem are increasing the model complexity and/or improving the feature selection.

Overfitting is the opposite of underfitting. It happens when a model performs well on the training partition but it fails to generalize. In other words, the model is too complex to properly extract the patterns on data. Sometimes, such models seem to memorize the training data. As a result, they present low bias and high variance. Possible solutions to this problem are to simplify the model and/or collect more data. In Figure 5 is illustrated the concepts of underfitting and overfitting.

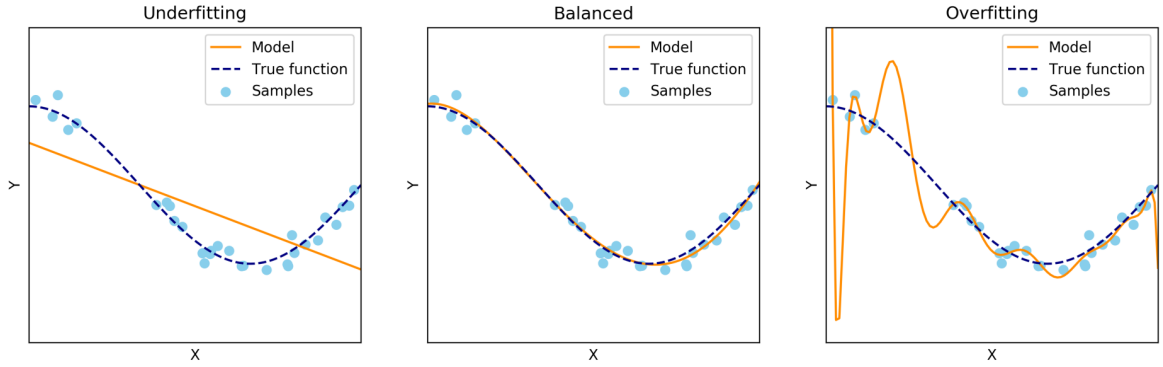


Figure 5 – Illustration of the concepts of underfitting and overfitting. In the first plot (from the left to the right) the model is too simple and cannot learn properly the data structure. In the middle, the model's complexity represents a good fit for the data. Lastly, in the last plot, the model is too complex that memorizes the training data. Adapted from scikit-learn documentation (Pedregosa et al., 2011).

2.1.4 Maximum Likelihood Estimation

In Statistical and Machine Learning, density estimation consists in estimating the Probability Density Function (PDF) of a random process based on observed data (James et al., 2013). There are different methods to estimate the data distribution. The Maximum Likelihood Estimation (MLE) is a general approach that is applied to fit several non-linear models in statistics (Myung, 2003; Smola; Vishwanathan, 2008). It is often considered the standard estimator for Machine Learning algorithms mainly because as the number of samples $N \rightarrow \infty$, the maximum likelihood estimate of a parameter converges to the true value of the parameter (Hastie; Tibshirani; Friedman, 2009; Goodfellow; Bengio; Courville, 2016).

Let us consider a data distribution $p_m(X;\Theta)$ parametrized by Θ. The idea behind the MLE algorithm is to select a value of Θ that maximizes the likelihood that the data would have been generated by $p_m(X;\Theta)$. In other words, the algorithm aims to approximate $p_m(X;\Theta)$ to the true data distribution $p_d(X)$. The maximum likelihood estimation for Θ

is defined as (Hastie; Tibshirani; Friedman, 2009):

$$\begin{aligned} \mathcal{L}(\Theta; X) &= \underset{\Theta}{\operatorname{argmax}}\ p_m(X; \Theta) \\ \mathcal{L}(\Theta; X) &= \underset{\Theta}{\operatorname{argmax}}\ \prod_{i=1}^{N} p_m(\mathbf{x}_i; \Theta) \end{aligned} \tag{2.4}$$

A mathematical trick to avoid the product of various probabilities, which may result in numerical underflow, is to take the logarithmic of the likelihood. This operation does not change the argmax value and convert the product into a sum (Goodfellow; Bengio; Courville, 2016):

$$\mathcal{L}(\Theta; X) = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_m(\mathbf{x}_i; \Theta) \tag{2.5}$$

In addition to the logarithmic operation, the argmax does not change when the cost function is rescaled. Thereby, we can divide it by N to get the function expressed as the expectation of the empirical distribution $p_d(X)$ (Goodfellow; Bengio; Courville, 2016):

$$\mathcal{L}(\Theta; X) = \underset{\Theta}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{x} \sim p_d} \left[\log p_m(\mathbf{x}; \Theta)\right] \tag{2.6}$$

The similarity between the model distribution $p_m(X, \Theta)$ and the empirical distribution $p_d(X)$ may be expressed by the Kullback–Leibler divergence (Kullback; Leibler, 1951) according to:

$$D_{KL}(p_d \,||\, p_m) = \underset{\Theta}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{x} \sim p_d} \left[\log p_d(\mathbf{x}) - \log p_m(\mathbf{x}; \Theta)\right] \tag{2.7}$$

Thus, one way to approximate $p_m(X, \Theta)$ to $p_d(X)$ is minimizing $\log p_m(\mathbf{x}; \Theta)$. In this sense, the term $\log p_d(\mathbf{x})$ in Equation 2.7 refers only to the data generation process and does not affect the model. Therefore, the maximum likelihood estimation may be achieved by minimizing only the right part of the Equation 2.8, which results in the following equation:

$$\mathcal{L}(\Theta; X) = -\mathbb{E}_{\mathbf{x} \sim p_d} \left[\log p_m(\mathbf{x}; \Theta)\right] \tag{2.8}$$

Observe that minimizing $D_{KL}(p_d \,||\, p_m)$ is exactly the same as maximizing Equation 2.6. In addition, minimizing the KL divergence between both distributions is equivalent to minimizing the cross-entropy between them. Therefore, any function defined as the negative log-likelihood is a cross-entropy between $p_m(X, \Theta)$ and $p_d(X)$ (Goodfellow; Bengio; Courville, 2016).

Thereby, for supervised learning problems, the MLE algorithm may also be applied to estimate the conditional distribution $p_m(Y \mid X; \Theta)$ – discussed in Section 2.1.2. In this case, the MLE is achieved as follows:

$$\mathcal{L}(\Theta; X, Y) = \underset{\Theta}{\operatorname{argmax}}\, \mathbb{E}_{\{\mathbf{x},y\}\sim p_d} \left[\log p_m(y \mid \mathbf{x}; \Theta)\right] \tag{2.9}$$

2.2 Artificial Neural Networks (ANN)

Artitificial Neural Networks (ANN) are a family of models that were inspired by the structure and function of the brain. Haykin (2010) defined an ANN as follows:

> *A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:*
>
> 1. *Knowledge is acquired by the network from its environment through a learning process.*
> 2. *Inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

The massive interconnections are composed of processing units known as artificial neurons (neurons for short). A group of neurons composes a layer. As the number of layers increases, the network becomes deep, which is the core idea of Deep Learning. There are several types of ANNs. As previously mentioned, it is a family of models. In this section, we introduce the fundamentals of Feedforward Neural Network (FNN) and the learning process.

2.2.1 Feedforward Neural Networks

The Feedforward Neural Network (FNN) is a type of neural network in which information flows through the layers without feedback. As any supervised learning algorithm, an FNN aims to learn how to approximate a function $f : X \to Y$ to solve a given problem. The basic unit of an FNN is the perceptron, which is described in the next section.

2.2.1.1 Perceptron

The perceptron is the simplest learnable processing unit of an FNN. It is defined as (Haykin, 2010):

$$f(\mathbf{x}) = \varphi\left(\sum_{i=1}^{m} x_i w_i + w_0\right) = \varphi\left(\mathbf{w}^T\mathbf{x} + w_0\right) \tag{2.10}$$

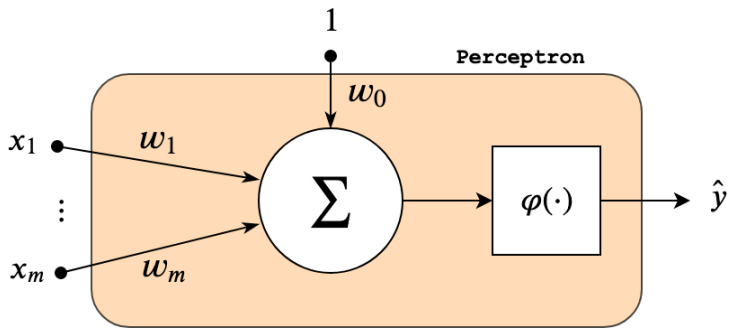


Figure 6 – Illustration of the perceptron, the basic unit of an FNN. As defined by Haykin (2010), the neuron's weights store the knowledge within the neuron.

where $\mathbf{x} = \{x_i, \cdots, x_m\}$ are the input data, $\mathbf{w} = \{w_1, \cdots, w_m\}$ are the neuron weights, w_0 is a special weight known as *bias*[1], $\varphi(\cdot)$ is the activation function, and the true output data y is estimated by $\hat{y} = f(\mathbf{x})$. Examples of activation functions are:

- Sigmoid or logistic function:

$$\sigma(u) = \frac{1}{1 + e^{-u}} \tag{2.11}$$

- Hyperbolic tangent:

$$\tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \tag{2.12}$$

- Softmax:

$$s(\mathbf{u}) = \frac{e^{u_k}}{\sum_k e^{u_k}} \tag{2.13}$$

- Rectified linear units (ReLU):

$$\text{ReLU}(u) = \max\{0, u\} \tag{2.14}$$

In summary, the perceptron is basically an affine transformation followed by a non-linear operation. Also, we may interpret the function $f(\mathbf{x})$ as the $p(y \mid \mathbf{x}; \Theta)$, where $\Theta = \{\mathbf{w}, w_0\}$ is the distribution parameters. Thus, a perceptron neuron learns when its weights is optimized for a given task. The perceptron is illustrated in Figure 6.

2.2.1.2 Multilayer Perceptrons (MLP)

A perceptron neuron outputs a single value $\hat{y}$. For multi-output tasks, we must use a group of neurons to map each output $\hat{\mathbf{y}} = \{\hat{y}_1, \cdots, \hat{y}_n\}$. This group of neurons is known as neural layer (layer for short) and is defined as:

$$\hat{y}_n = f_n(\mathbf{x}) = \varphi\left(\mathbf{w}_n^T \mathbf{x} + w_{0_n}\right) \tag{2.15}$$

[1] In this case, bias is a shift operator and is not related to the concepts introduced in Section 2.1.3.1

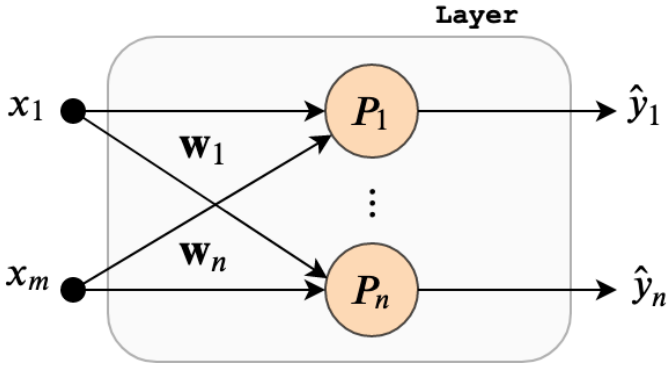


Figure 7 – Illustration of a neural layer composed of $\{P_1, \cdots, P_n\}$ perceptrons. Each perceptron has its own set of weights and outputs a single prediction. For simplicity, the *bias* was omitted.

In this case, each neuron has its group of weights, represented by $W = \{\mathbf{w}_1, \cdots, \mathbf{w}_n\}$, and the predicted outputs is described as $\hat{\mathbf{y}} = f(\mathbf{x})$, which is composed of multiples output functions $\{f_1(\mathbf{x}), \cdots, f_n(\mathbf{x})\}$. In Figure 7 is illustrated a neural layer.

In order to increase the network complexity, we may stack a sequence of layers to form a Multilayer Perceptron (MLP) as illustrated in Figure 8. Mathematically, an MLP can be interpreted as a sequence of composite functions that are parameretized by the weights of the network:

$$f(\mathbf{x}) = \varphi_L \left(W_L^T \left[\varphi_{L-1} \left(W_{L-1}^T \left[\cdots \varphi_1 \left(W_1^T \mathbf{x} + \mathbf{w}_{0_1} \right) \right] \mathbf{w}_{0_{L-1}} \right) \right] + \mathbf{w}_{0_L} \right) \tag{2.16}$$

where L is the number of layers. In this sense, layer 1 provides the output for layer 2, layer 2 for layer 3, and so on. In addition, the first layer is the input, the last layer is the output, and the remaining ones are the hidden layers. The number of hidden layers determines the model's depth.

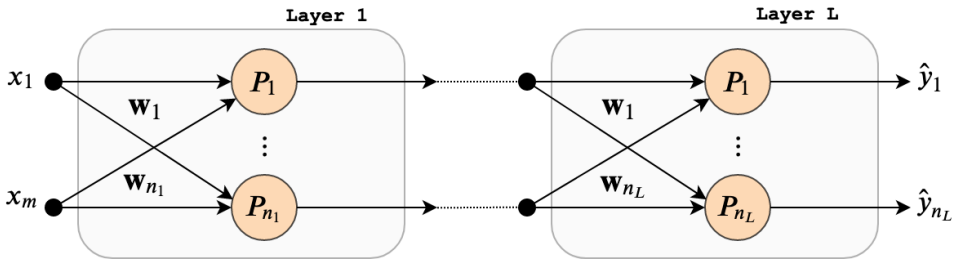


Figure 8 – Illustration of a Multilayer Perceptron (MLP) composed of L layers. As it is a feedforward network, information flows from layer 1 to Layer L, which represents the output predictions. Each layer may have a different number of neurons.

According to the universal approximation theorem (Cybenko, 1989) an MLP with at least one hidden layer – and any non-linear activation function – can approximate any

continuous function. However, the theorem does not inform how large – in terms of the number of neurons in the hidden layer – the network must be. In addition, even in a hypothetical situation in which we know this number, the learning algorithm can fail to fit the model (Bengio et al., 2009). In practice, MLPs with more than four hidden layers rarely present better generalization than those ones with fewer layers (Li et al., 2019a).

2.2.2 Training a Feedfoward Neural Network

Training a neural network is the task of finding the values for the weights connections that best represent the observed data. Thereby, it is basically an optimization problem with respect to a cost function. There are different ways to train an FNN such as using evolutionary-based algorithms (Miller; Todd; Hegde, 1989; Salimans et al., 2017), simulated annealing (Rere; Fanany; Arymurthy, 2015), and colony optimization (Mavrovouniotis; Yang, 2015). However, the most successful approach is the backpropagation (BP) algorithm (Rumelhart; Hinton; Williams, 1986), which is based on gradient descent optimization.

2.2.2.1 Gradient descent optimization

Let us consider a function $f_c(\mathbf{z})$ in which we aim to minimize – this function is known as cost function. The vector containing all partial derivatives of $f_c(\mathbf{z})$ with respect to a point $z_i \in \mathbf{z}$ is named gradient and is defined as:

$$\nabla_{\mathbf{z}} f(\mathbf{z}) = \frac{\partial f_c(\mathbf{z})}{\partial z_i} \tag{2.17}$$

The gradient vector points towards the maximum value of f_c. Thus, the idea of the gradient descent optimization is to iteratively move $\mathbf{z}$ in small steps using the negative of the gradient (Cauchy, 1847):

$$\mathbf{z}^{\text{new}} = \mathbf{z} - \alpha \nabla_{\mathbf{z}} f(\mathbf{z}) \tag{2.18}$$

where α is the learning rate, a hyperparameter that indicates the step size.

For functions in which both input and output are vectors, i.e., $\mathbf{f}_c : \mathbb{R}^n \to \mathbb{R}^m$, we need to compute a matrix $J \in \mathbb{R}^{m \times n}$ of partial derivatives:

$$J_{i,j} = \frac{\partial f_c^i(\mathbf{z})}{\partial z_j} \Rightarrow \begin{bmatrix} \frac{\partial f_c^1(\mathbf{z})}{\partial z_1} & \cdots & \frac{\partial f_c^1(\mathbf{z})}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_c^m(\mathbf{z})}{\partial z_1} & \cdots & \frac{\partial f_c^m(\mathbf{z})}{\partial z_n} \end{bmatrix} \because i = 1, \cdots, m \text{ and } j = 1, \cdots, n \tag{2.19}$$

This matrix is called Jacobian matrix and is quite useful to compute partial derivatives in neural networks learning algorithms.

2.2.2.2 Cost function

In order to apply an optimization algorithm to train an MLP, we need to define a cost function for the network – most of the time referred to as loss function. Let us consider a supervised problem with $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ inputs and $Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$ outputs, where N is the number o samples. The loss function measures the expected error between the true output Y and the estimated network output $\hat{Y} = f(X)$. The simplest way to compute this error is through the mean squared error (MSE):

$$\mathcal{L}(W) = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2 \tag{2.20}$$

Observe that it is only a different formulation of Equation 2.3.

As previously discussed, we may interpret an MLP as the conditional distribution $p(\mathbf{y} \mid \mathbf{x}; \Theta)$. From this perspective, we may use the cross-entropy between the true and the estimated conditional distribution – see Section 2.1.4 – as loss function:

$$\begin{aligned} \mathcal{L}(\Theta; X, Y) &= -\mathbb{E}_{\{X,Y\}\sim p_d}\left[\log p_m(Y \mid X; \Theta)\right] \\ \mathcal{L}(\Theta; X, Y) &= -\tfrac{1}{2}\textstyle\sum_{i=1}^{N} \mathbf{y}_i \log \hat{\mathbf{y}}_i \end{aligned} \tag{2.21}$$

where the distribution parameters Θ is composed of the network's weights and the bias. Thus, we can apply the gradient descent update rule, described in Equation 2.18, to the cross-entropy loss function:

$$\Theta^{\text{new}} = \Theta - \alpha\nabla_{\Theta}\mathcal{L}(\Theta; X, Y) \tag{2.22}$$

2.2.2.3 Backpropagation algorithm

In the gradient descent algorithm, we assume that the gradient computation of the loss function is done analytically. However, as shown in Section 2.2.1.2, an MLP consists of many functions chained together. As a result, evaluating such a sequence of functions becomes computationally expensive as the network increases. To deal with this problem, Rumelhart, Hinton and Williams (1986) proposed the backpropagation, a breakthrough solution to efficiently compute the gradients through the network that is still used today. In summary, the backpropagation is able to compute the gradient over the network's loss function with respect to every single parameter in Θ. Next, it applies the gradient descent update rule until the network converges to a solution (Géron, 2019). It is worth noting that backpropagation is a method to compute gradients. The network's learning is performed by another algorithm such as gradient descent and its variations (Goodfellow; Bengio; Courville, 2016).

The main idea of backpropagation algorithm is to propagate the error, obtained by the network's loss function, from the output layer back to the input layer to compute the gradient using the calculus chain rule. For example, let us consider two functions $\nu_1 : \mathbb{R}^m \to \mathbb{R}^n$ and $\nu_2 : \mathbb{R}^n \to \mathbb{R}$. If $\mathbf{u} = \nu_1(\mathbf{v})$ and $a = \nu_2(\mathbf{u})$, the chain rule is defined as:

$$\frac{\partial a}{\partial v_i} = \sum_j \frac{\partial a}{\partial u_j} \frac{\partial u_j}{\partial v_i} \tag{2.23}$$

In vector notation:

$$\nabla_{\mathbf{v}} a = \left(\frac{\partial \mathbf{u}}{\partial \mathbf{v}}\right)^T \nabla_{\mathbf{u}} a \tag{2.24}$$

where $\frac{\partial \mathbf{u}}{\partial \mathbf{v}}$ is the Jacobian matrix of ν_1. This operation is applied to compute the gradient through the network.

Let us consider the MLP with L layers illustrated in Figure 8. In addition, let $\mathbf{a}^l$ and $\mathbf{u}^l$ be the vector of pre-activations and activations functions, respectively. The gradient of the loss function with respect to parameters Θ^l for the l^{th} layer is computed as (Zhou; Greenspan; Dinggang, 2017):

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta^l} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^L} \frac{\partial \mathbf{a}^L}{\partial \mathbf{a}^{L-1}} \cdots \frac{\partial \mathbf{a}^{l+2}}{\partial \mathbf{a}^{l+1}} \frac{\partial \mathbf{a}^{l+1}}{\partial \mathbf{a}^l} \frac{\partial \mathbf{a}^l}{\partial \mathbf{u}^l} \frac{\partial \mathbf{u}^l}{\partial \Theta^l} \tag{2.25}$$

Observe that $\hat{\mathbf{y}} = \mathbf{a}^L$. In this sense, $\frac{\partial \mathcal{L}}{\partial \mathbf{a}^L}$ correspond to the gradient of the predicted output. Also, it is possible to observe that the gradient value is propagated from the output layer through the network in the form of $\frac{\partial \mathbf{a}^{l+1}}{\partial \mathbf{a}^l}$ in association with $\frac{\partial \mathbf{a}^l}{\partial \mathbf{u}^l} \frac{\partial \mathbf{u}^l}{\partial \Theta^l}$. The derivative $\frac{\partial \mathbf{a}^{l+1}}{\partial \mathbf{a}^l}$ may be computed as follows:

$$\begin{aligned} \frac{\partial \mathbf{a}^{l+1}}{\partial \mathbf{a}^l} &= \frac{\partial \mathbf{a}^{l+1}}{\partial \mathbf{u}^{l+1}} \frac{\partial \mathbf{u}^{l+1}}{\partial \mathbf{a}^l} \\ \frac{\partial \mathbf{a}^{l+1}}{\partial \mathbf{a}^l} &= \frac{\partial \mathbf{a}^l}{\partial \mathbf{u}^l} \left(\Theta^l\right)^T \end{aligned} \tag{2.26}$$

Once the gradient of the loss function with respect to each weights and bias is computed, the gradient descent update rule is applied.

2.2.3 Gradient descent variants

The gradient descent algorithm described in Section 2.2.2.1 has different variants that improve its performance in terms of effectiveness and convergence time. In this section, we describe the Stochastic Gradient Descent (SGD) and Momentum.

2.2.3.1 Stochastic Gradient Descent (SGD)

The original gradient descent algorithm computes the gradient of the loss function $\nabla_\Theta \mathcal{L}(\Theta; X, Y)$ considering all samples X and Y in the training set to perform only one update. Since in Deep Learning large datasets are required to achieve good generalization, this computation becomes computationally expensive, and sometimes, unfeasible since the dataset may not fit on memory.

Stochastic Gradient Descent (SGD) is an extension of the original method that computes the gradient using a small set of samples called mini-batches. The method relies on the fact that the gradient is an expectation that can be approximately estimated by the mini-batches (Goodfellow; Bengio; Courville, 2016). In supervised learning, a mini-batch is defined as $B = \{(X_1, Y_1), \cdots, (X_{bs}, Y_{bs})\}$ drawn from the training partition, where bs is the batch size. Usually, it assumes values ranging from 1 to a few hundred. For example, if a training set has 100 samples, we may set a batch size equal to 25 to get 4 mini-batches. After this operation, the gradient update rule is defined as:

$$\Theta^{\text{new}} = \Theta - \alpha \nabla_\Theta \mathcal{L}(\Theta; X_{\{1,\cdots,bs\}}, Y_{\{1,\cdots,bs\}}) \tag{2.27}$$

The update rule is performed to all mini-batches in the training set – this is known as an epoch. After every epoch, the samples are shuffled and mini-batches are generated again. The number of epochs required to reach convergence depends on the problem and the size of the training set.

In addition to be much faster than the original method (Ruder, 2016), the SGD allows online learning, i.e., it is possible to update the network with new samples on-the-fly. This is a very desirable feature since the training samples may not be available at the same time. Also, using SGD, we may train an MLP using a dataset with millions of samples, since the update is done using only bs samples. For these reasons, SGD combined with backpropagation is one of the algorithms of choice when training neural networks. In Algorithm 1 is described the pseudocode of the SGD method.

Algorithm 1: *Stochastic Gradient Descent (SGD)*

Input:
    Training set (Tr): X and Y;
    Loss function: $\mathcal{L}(\Theta; X, Y)$;
    Hyper-parameters: bs, α, and number of epochs (ep);
while $j < ep$ **do**:
    Shuffle Tr and compute the mini-batches;
    while $i <$ size(Tr) $\div$ bs **do**:
        $\Theta^{\text{new}} = \Theta - \alpha \nabla_\Theta \mathcal{L}(\Theta; X_i, Y_i)$;
return Θ.

2.2.3.2 Momentum

Although SGD is effective, it may present slow convergence for multimodal loss functions. Momentum helps to accelerate the SGD learning (Qian, 1999). Formally, it introduces a new variable $\mathbf{v}$ that represents velocity, i.e, the direction and speed at which the parameters move through parameter space (Ruder, 2016; Goodfellow; Bengio; Courville, 2016):

$$\mathbf{v}^{\text{new}} = \gamma\mathbf{v} - \alpha\nabla_{\Theta}\mathcal{L}(\Theta; X_i, Y_i) \tag{2.28}$$

where γ is the momentum coefficient, a hyperparameter ranging from 0 to 1 that controls the contributions of the previous gradient decay. The network parameters' update rule is now defined as:

$$\Theta^{\text{new}} = \Theta + \mathbf{v}^{\text{new}} \tag{2.29}$$

To update Algorithm 1, we only need to replace the line 8 with Equations 2.28 and 2.29. The remaining lines are still the same.

In Newtonian mechanics, momentum is mass times velocity. In the momentum learning algorithm context, which is obviously inspired by physics concepts, the gradient is a force that moves particles on parameter space. Ruder (2016) suggested the following analogy:

> *Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way. The same thing happens to our parameter updates: the momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, we gain faster convergence and reduced oscillation.*

2.2.4 Regularization

In statistics and machine learning, regularization is any procedure that aims to increase the generalization of a deep model by introducing constraints or adding new information into the model. Typically, a deep neural network may have millions of trainable parameters and can easily overfit for most datasets (Géron, 2019). Regularization is a way to tackle this problem. There are several regularization methods. In this section, we briefly describe the weight decay, dropout, batch normalization, and early stopping.

2.2.4.1 Weight decay

Weight decay, also known as ℓ_2 regularization, is probably the most common technique for parametric regularization in Machine Learning. In brief, it is a regularization method that stimulates the model to pursue small weights by adding a penalty term into the cost function (Krogh; Hertz, 1992). Essentially, small weights may result in a more stable model in which small input changes will not cause a significant change in the output (Brownlee, 2018).

Let us consider an arbitrary cost function $\mathcal{L}(\mathbf{w})$. In order to ensure small weights to $\mathbf{w}$, we define a new cost function defined as:

$$\mathcal{L}^{\mathrm{R}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{2.30}$$

where $\|\mathbf{w}\|$ is the norm of the weights and λ is the regularization constant, a non-negative hyperparameter. Now, if the weights get too large, the norm will increase and, consequently, the cost function. Thus, the learning algorithm will focus on minimizing the norm, which results in smaller weights (Zhang et al., 2020). To control the trade-off between the model cost function and the penalty term, it is introduced a non-negative hyperparameter λ known as regularization constant – in most machine learning libraries, this parameter and the whole method is interchangeably named weight decay. Essentially, the higher the value of λ, the more the learning algorithm constrains the weights. When $\lambda = 0$, no constrain is applied.

2.2.4.2 Dropout

Another common technique to regularize neural networks is Dropout (Srivastava et al., 2014). Basically, during the training phase, every neuron has a probability p of being temporarily turned-off for a single training step, i.e., the neuron is ignored during this step – see Figure 9. In the next training step, it may be activated again, depending on p, which is a hyperparameter called dropout rate (Géron, 2019). Thereby, Dropout provides a quite cheap way to simulate different models being trained in parallel. As a result, it tends to slow down convergence. Nonetheless, despite simple, Srivastava et al. (2014) proved that this method significantly reduces overfitting and improves generalization, which justifies the extra time.

2.2.4.3 Batch normalization

Training deep neural networks with backpropagation may result in the vanishing/exploding gradient problem. In summary, the vanishing/exploding gradient happens when the gradients get too small/big that the algorithm either does not update the weights anymore or the weights get huge values. In both cases, the model does not converge to a proper

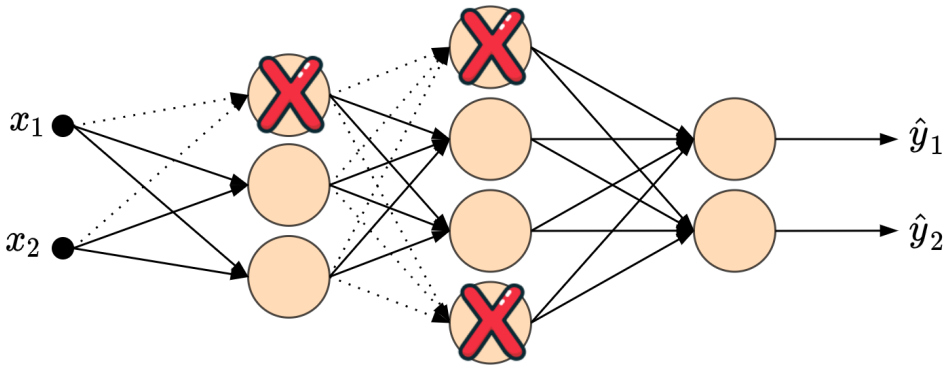


Figure 9 – An example of Dropout regularization in a small neural network. The red X means that the neuron is turned-off for the current step. Note that Dropout is not applied to the output layer.

solution (Géron, 2019). The main goal of the batch normalization (Ioffe; Szegedy, 2015) is to deal with this problem, however, it also has a regularization effect that sometimes makes dropout unnecessary (Goodfellow; Bengio; Courville, 2016).

The main idea behind the batch normalization method is that different layers may learn at different speeds. Thus, the method normalizes batches of inputs of individual layers of a deep model according to the following equation (Ioffe; Szegedy, 2015):

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta \tag{2.31}$$

where $\mathbf{x}$ is the mini-batch of data, $\hat{\mu}$ and $\hat{\sigma}$ are the mean and standard deviation of $\mathbf{x}$, respectively, and γ and β are the scaling and offset coefficients. The method works as follows: for each training step, the mini-batch is normalized to get zero mean and unit variance. Next, this value is re-scaled according to γ and β. This operation is important to ensure that the activation magnitudes among the layers do not diverge. With large enough mini-batch, the method is effective to speed up learning and improve generalization (Zhang et al., 2020). It is important to note that γ and β must be jointly included to parameters Θ to be learned during backpropagation.

2.2.4.4 Early stopping

Early stopping is a quite simple approach to regularize iterative learning algorithms. The basic idea is to stop the training phase according to the error of the validation partition. In other words, when the training error keeps decreasing and the validation error does not, it means the model is overfitting and we should stop training (Zhang et al., 2020; Géron, 2019). Two common thresholds we can use to stop training a model are described as follows:

- **Minimum validation error**: we can set a minimum error threshold and, if the validation error reaches it, we stop training the model.
- **Number of epochs without improvement**: we can set a maximum number of epochs without improvement on the validation error and, if the model reaches it, we stop training the model.

Usually, the number of epochs without improvement is a more common threshold since knowing the minimum validation error upfront may not be feasible. In any case, early stopping is such a simple and efficient approach that is very desired when training any kind of model.

2.3 Convolutional Neural Networks (CNN)

Convolutional Neural Network (CNN) (LeCun et al., 1989; LeCun; Bengio et al., 1995; LeCun et al., 1998) is a special type of Artificial Neural Network that was designed to handle grid-like data. Different architectures of CNNs have been successfully applied to many real-world problems such as Natural Language Processing (NLP), automatic video classification, voice recognition, self-driving car, among many others (Rawat; Wang, 2017; Géron, 2019; Shen et al., 2019). Nowadays, this is the most successful deep learning method to learn visual features from images (Rawat; Wang, 2017). As the name suggests, a CNN is a network that applies the convolution operator, instead of general matrix multiplication, in at least one of their layers (Goodfellow; Bengio; Courville, 2016). The network is designed to exploit extensive weight-sharing in order to reduce computational complexity, which significantly differs it from traditional MLPs. In this section, we first present the motivation and inspiration of CNNs models. Next, we describe the fundamental concepts regarding this model.

2.3.1 Local receptive fields in the visual cortex

The convolutional neural network is a direct inspiration from studies of the brain's visual cortex, the part of brain responsible to process visual information. This part is composed of different areas in the brain and each area has a specific function that hierarchically process the vision information (Trappenberg, 2009). Hubel and Wiesel (1959) demonstrated that neurons in the visual cortex have a small local receptive field, i.e., they respond only to visual inputs from a limited region of the visual field. The receptive fields of different neurons may overlap to cover the whole visual field, as illustrated in Figure 10. They also observed that neurons may have different sizes of receptive fields and some of them only react to more complex patterns that are a composition of other lower-level

patterns. This hierarchical architecture is able to identify complex patterns in any part of the visual field (Géron, 2019).

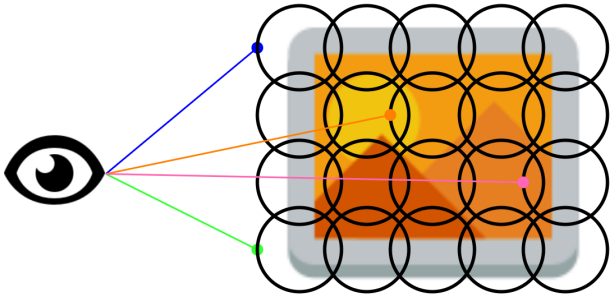


Figure 10 – Illustration of the local receptive fields in the visual cortex. Neurons in this part of the brain reacts only for limited regions of the visual field (Géron, 2019).

The concepts of the visual cortex inspired the development of the CNNs, which was first introduced as Neocognitron (Fukushima, 1980) and gradually evolved to the modern version that is widely used nowadays. Basically, the stack of layers of a CNN aims to simulate the hierarchical architecture of the cortex. In addition, the local receptive field concept is replicated in the convolutional layers through the kernels, which have spatial connectivity and shared weights. These characteristics allow CNNs to deal with high-dimensional data, which is a fundamental advantage compared to the MLPs.

2.3.2 CNN layers

Essentially, a standard CNN is composed of three types of layers: convolutional, pooling, and fully-connected layers (LeCun et al., 1989; LeCun et al., 1998). In this section, we describe the convolutional and pooling one. The fully-connected layer is the same feedfoward network described in Section 2.2.1.

2.3.2.1 Convolutional layer

The convolutional layer is the most important layer of the network. It is based on the convolution operator, a mathematical operation on two functions, let us say g and h, that produces a third function that can be interpreted as how h modifies g:

$$(g \circledast h)(t) = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)\, d\tau \tag{2.32}$$

where g and h are two continuous function in which the integral is defined. Considering the discrete case:

$$(g \circledast h)[n] = \sum_{m=-\infty}^{\infty} g[m]h[n-m] \tag{2.33}$$

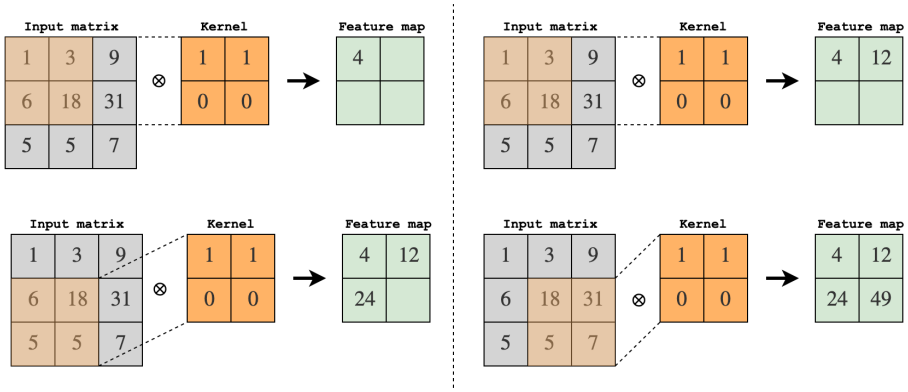


Figure 11 – An example of the convolution between a 3×3 input matrix and a 2×2 kernel. In this example, the kernel convolves only in defined values of the image. The symbol $\otimes$ represents the dot product between the selected values.

In the convolutional network terminology, the function g is a multidimensional array of data called input – for example, an image. The function h is a multidimensional array of learnable parameters called kernel or filter. Lastly, the convolution result is another multidimensional array of data named feature maps. These multidimensional arrays are often named tensors (Goodfellow; Bengio; Courville, 2016).

Let us consider a 2D input matrix I and a 2D kernel K. Assuming that every point that is not defined in these matrices is zero, it is possible to implement Equation 2.33 over a finite number of array elements:

$$F_{\text{map}} = (I \circledast K)[i, j] = \sum_{n_1} \sum_{n_2} I[n_1, n_2] K[i - n_1, j - n_2] \tag{2.34}$$

The convolution between an input matrix, which could be an image, and a kernel is illustrated in Figure 11. This operation is commonly used in image processing algorithms for edge detection, blurring, sharpen, among others (Solem, 2012). It means that different kernels may extract different features from the image. Nonetheless, to perform these image processing operations, the values of the kernels must be known beforehand. In CNN, these values are the weights and are obtained using a learning process. After training, feature maps generated by the convolution contain important characteristics that the kernel learned to extract from the image. In a nutshell, the idea of local kernels follows the concept of the local receptive fields in the visual cortex – in fact, many authors use kernels and local receptive fields interchangeably (Zhang et al., 2020).

Parameters of a convolutional layer

The example illustrated in Figure 11 shows an image convolved by only one kernel. Nevertheless, a convolutional layer may contain many kernels and feature maps. Actually,

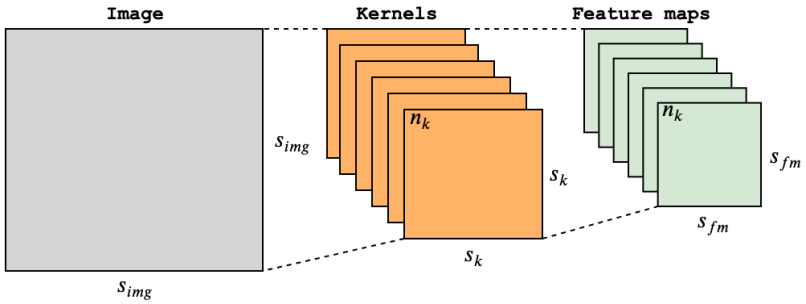


Figure 12 – Illustration of a convolution layer containing six kernels. Each kernel output a 2D feature map that are stacked at the end. The shape of the feature maps are defined by Equation 2.35.

there are four parameters that impact on the size and number of feature maps (Li et al., 2019a):

- **Number of kernels** (n_k): it corresponds to the number of kernels that convolves over the input matrix. Each kernel produces one feature map, which may be interpreted as a stack of 2D matrices – as illustrated in Figure 12. For this reason, this number is also known as the layer's depth.
- **Size of kernels** (s_k): it corresponds to the shape of the kernel. Typically, all kernels within the same layer have the same shape $[s_k, s_k]$.
- **Stride** (str): it corresponds to the number of pixels by which the kernel is slid over the input matrix. When stride is 1, the kernel moves one pixel at a time; when it is two, the kernel moves two pixels at a time; and so on. It is worth noting that a larger stride will produce smaller feature maps – see Equation 2.35.
- **Zero-padding** (zp): it consists of wrapping the input matrix with zeros around the border. This operation also allows us to control the feature maps' size – see Equation 2.35. The parameter zp corresponds to the number of lines of zeros surrounded the input matrix.

The equation to compute the feature maps spatial size of a convolutional layer is defined as:

$$s_{fm} = \frac{s_{img} - s_k + 2zp}{str} + 1 \tag{2.35}$$

Applying this equation to the example illustrated in Figure 11, we have a 3×3 input matrix, a 2×2 kernel, stride equat to 1, and no zero-padding. Therefore, the size of the feature map is $\frac{3-2+0}{1} + 1$, which results in a 2×2 matrix.

Non-linearity

After a convolution layer, usually is applied an activation function to add non-linearity into the layer. Theoretically, this function can be, for example, a sigmoid or ReLU, which were described in Section 2.2.1.1. In practice, Bengio (2012) showed that ReLU generally works better for convolutional layers. Thus, it is straightforward to apply the ReLU to a set of feature maps F_{map}:

$$\tilde{F}_{\text{map}} = ReLU(F_{\text{map}}) \tag{2.36}$$

In brief, this operation removes all negative pixels from the map.

Properties of a convolutional layer

The convolutional layer has three important properties that help to explain why this model is the most successful one to handle large amounts of data, in particular, images. They are: sparse interactions, parameter sharing, and equivariance.

In standard neural network layers every output neuron connects with every input neuron. As described in Equation 2.10, it is defined in terms of matrix multiplication. For a large matrix input, such as an image, connecting a neuron to each pixel is computationally expensive, and most of the time, unfeasible in terms of computational memory. In convolutional layers, the kernel is smaller than the input matrix, which results in sparse interactions. It means that the input matrix may have thousands of values, but only a small portion is assessing to detect relevant features. In addition, convolutional layers use the same weights in a kernel to generate multiples outputs that compose the feature maps. In other words, each part of the kernel is convolved through the whole matrix input. This characteristic is known as parameter sharing since there is no weight for each value of the input matrix. Sparse interactions along with parameter sharing substantially reduce the number of parameters and operations when compared to a standard MLP layer. As a result, this layer is faster and requires less computational memory to run (Goodfellow; Bengio; Courville, 2016).

Another important property of convolutinal layers is the equivariance to translation. Mathematically, a function is equivariant if a change in the input results in a change in the output in the same way. In the convolution operation, if we move an object in the input image, its representation in the feature maps will move in the same way. This is particularly useful to identify small variations in images pattern. However, it is important to note that convolution is not naturally equivariant to all transformations. For example, we need to provide other techniques to identify changes in scale or rotations in an input image – we discuss this in Section 2.3.4.

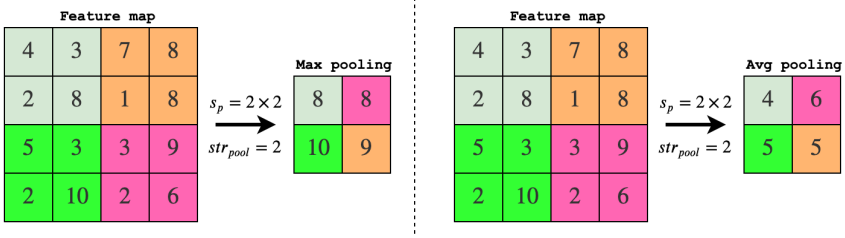


Figure 13 – Example of the pooling operation in a 4×4 feature map using MaxPool (left) and AvgPool (right) with pooling stride equal to 2 and spatial neighborhood equat to 2×2.

2.3.2.2 Pooling layer

The pooling operator (also known as sub-sampling) is a function that summarizes a region of the feature maps by means of a pre-defined statistics. First of all, we need to define two hyperparameters: the size of spatial neighborhood (s_{pool}) and the pooling stride (str_{pool}). Next, every region $[s_{pool}, s_{pool}]$ is summarized by str_{pool} points of spacing. This function can be the maximum (MaxPool) or average (AvgPoll) of values, as illustrated in Figure 13.

A clear benefit of the pooling layer is reducing spatial size of feature maps and, consequently, the number of parameters and computation in the network. As we can observe from Figure 13, the pooling operation reduces in 75% the size of the feature map. In addition, since the pooling operation summarizes activations over a whole neighborhood, it helps the network to deal with small translations and distortions in the input. However, it is worth noting that a convolutional network may work properly without any pooling layer. In fact, some works suggest that the pooling operation waste important features that may be important in pattern recognition (Sabour; Frosst; Hinton, 2017). Other works simulate a pooling layer by adding repeated convolutional layers before output a feature map (Zagoruyko; Komodakis, 2016; Huang et al., 2017b). In the end, there is no consense regarding this subject, since we also have successful networks that apply pooling in their architectures (Simonyan; Zisserman, 2014; Szegedy et al., 2015; Sandler et al., 2018).

2.3.3 Connecting the layers

A standard covolutional network consists of a stack of convolution, activation function, and pooling layers. These layers are the basic building blocks of CNNs. After a predefined number of layers, the last feature maps are forwarded to a fully-connected layer, which is a standard feedforward network. The name fully-connected refers to the fact that in MLPs all inputs are connected to all neurons, as discussed in Section 2.3.2.1. In Figure 14 is shown a typical CNN architecture. As we can see, convolution layers, followed by a non-linearity function, produce the feature maps. Pooling layers reduce these maps that work as input to the next layer. The last feature map is flattened and becomes the input

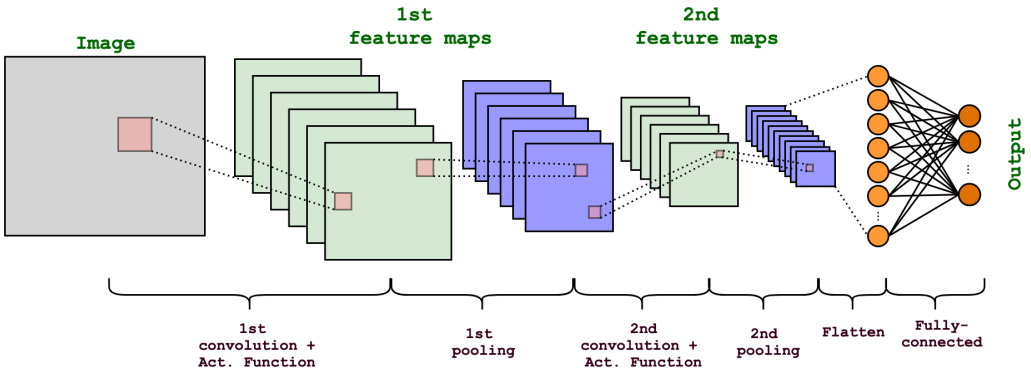


Figure 14 – A typical CNN architecture with two stacked layers performing convolution, activation function, and pooling. The feature maps computed in the last layer are flattened and acts as inputs to the fully connected layer. Lastly, the fully-connected layer provides the network's predictions.

to the fully-connected network, which produces the CNN's output. Usually, it is applied the softmax function – see Equation 2.13 – in the output layer, since it forces the output array to lie within the interval 0 to 1 and sum up 1, i.e., each output neuron may be interpreted as a probability.

Essentially, the fully-connected works as a classifier while the remaining layers extract features from the image. The architecture organization, i.e, sequence of layers, number of convolutions, number of fully-connected layers, and layers itself, are network hyperparameters. However, it is known that deeper models, i.e, models with a high number of convolution layers, perform better than the shallow ones (Goodfellow; Bengio; Courville, 2016). Successful CNN architectures in real-world problems may present more than one hundred layers (He et al., 2016; Simonyan; Zisserman, 2014; Szegedy et al., 2015). Usually, kernels of the first layers learn to detect basic features such as borders, shapes, corners, etc. As the network gets deeper, the kernels learn higher concepts like an eye in a person's face. This is the essence of hierarchical learning, each layer learns different concepts (Bengio, 2012).It is worth mentioning that it is possible to attach a different classifier into convolutional networks. For example, Niu and Suen (2012) used a Support Vector Machines (SVM) as a classifier instead of a feedforward network.

2.3.4 Techniques to improve CNNs performance

As described in Section 2.3.2.1, CNNs may present problems to deal with changes in scale or rotations in an input image. In addition, a known issue with this model is that it needs a large number of images to generalize well (Goodfellow; Bengio; Courville, 2016; Chollet, 2018; Géron, 2019). In this section, we describe data augmentation and transfer learning.

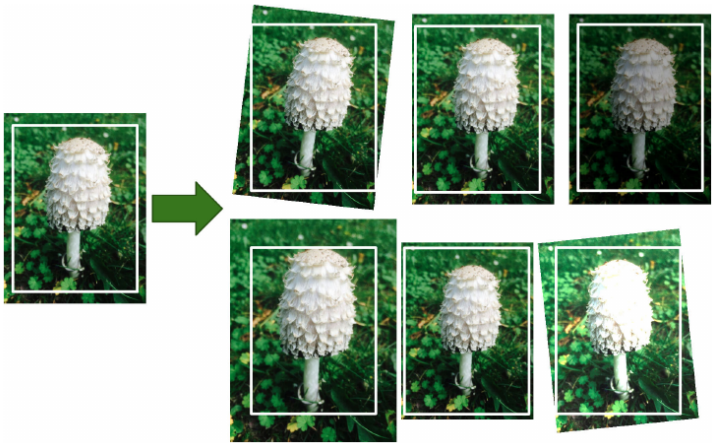


Figure 15 – Example of data augmentation on an image sample. As we can note, six samples are generated from the same image. Source: (Géron, 2019)

2.3.4.1 Data augmentation

Data augmentation is a technique to enlarge and diversify the number of samples of dataset. It consists of generating artificial samples based on the original ones. In the case that the samples are images, the new instances are generated using traditional image processing operations, such as re-scale, rotations, shearing, zooming, cropping, modifying brightness level, among others. As illustrated in Figure 15, these operations can be combined. In addition, they are carried out randomly.

Data augmentation plays an important role by introducing diversity into the data. These new samples make the model more tolerant to variations (Géron, 2019). In addition, it contributes to avoiding overfitting. As a consequence, it is a basic technique to perform when using CNNs. It is noteworthy that these operations cannot disturb the image at the point to make it unrecognizable; If so, the model learns concepts that are not related to the original data, which contributes to reduce performance.

2.3.4.2 Transfer learning

Collecting and annotating data may be a challenging task, in particular for medical problems. As a result, many problems do not have large datasets available to be used in deep learning models. In addition, data augmentation may not be enough to ensure appropriate performance. In this context, the idea of transfer learning (also known as knowledge transfer) arises. Basically, it involves training the model to another large dataset and then using the same weights as a starting point to a problem with fewer data available.

It is quite common pre-train a model using ImageNet (Deng et al., 2009) – a dataset that contains over 14 millions samples – and then finetune the same model for another problem. For example, Perez et al. (2018) show that performing this procedure helps to leverage performance for melanoma classification, even though images on ImageNet

are significantly different from skin lesions. Recently, Raghu et al. (2019) concluded that recycling the first layers is more important than the deeper ones on transfer learning for medical imaging applications. The hypothesis is because the first layers learn to extract common features, e.g., borders, that may be useful in different tasks. In any case, similar to data augmentation, transfer learning is a widely used technique in deep learning and is strongly recommended to handle problems that have small datasets.

3 Combining image and meta-data features

In Information Fusion (IF) the term multimodal fusion or heterogeneous fusion stands for the task of combining different types of data obtained from different sources (Atrey et al., 2010; Lin et al., 2016). Over the past few years, multimodal fusion has been actively applied to multimedia analysis. For example, Ortega et al. (2019) proposed a model to combine audio, video, and text data to perform human emotion recognition. The rationale behind using such a method is that combining heterogeneous data may provide complementary information as well as increase effectiveness in a decision-making pipeline (Atrey et al., 2010). There are different strategies to perform multimodal fusion, nonetheless, the most widely used is the feature level fusion, i.e., when features extracted from each type of data are combined using a predefined methodology. This is the off-the-shelf method applied to multimedia analysis that involves image processing.

Many image processing tasks such as disease classification, face recognition, image retrieval, and object identification are challenging because the difference in the images' visual pattern is quite small. This may be caused by different factors, for example, noise, viewpoint, and data variance (Lin; RoyChowdhury; Maji, 2015). In order to improve performance and increase robustness of methods that deal with these tasks, it is a common practice combine different features extracted from different algorithms or obtained from different sources of data (Nanni et al., 2016; Kharazmi et al., 2018; Liu et al., 2019; Ardila et al., 2019; Kabbai; Abdellaoui; Douik, 2019; Pacheco; Krohling, 2020a).

Several methods have been proposed to combine image features extracted from Convolutional Neural Networks (CNNs) and handcrafted features, i.e., those features extracted using traditional image processing algorithms – to extract shapes, color, or texture – or other computer vision methods such as Fisher Vector (Perronnin; Sánchez; Mensink, 2010), Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), Speed up Robust Feature (SURF) (Bay et al., 2008), and Vector of Locally Aggregated Descriptors (VLAD) (Jégou et al., 2010). To name a few, Arroyo and Zapirain (2014) and Majtner, Yildirim-Yayilgan and Hardeberg (2016) combined convolutional and hancrafted features to improve skin cancer classification, Nanni et al. (2016) for music genre classification, and Nguyen et al. (2018) for face detection. Other works, such as (Lin; RoyChowdhury; Maji, 2015), (Arroyo et al., 2016), (Liu et al., 2017), and (Ardila et al., 2019), proposed the combination of image features extracted from different CNN architectures. In both cases, the most common way to combine the extracted features are through feature concatenation (Nanni et al., 2016; Wang et al., 2017; Akram et al., 2018; Pogorelov et al., 2018; Li et al., 2018; Hai et al., 2019). A general example of a feature concatenation method is illustrated in Figure 16. Basically, the idea of this method is to attach all extracted features within

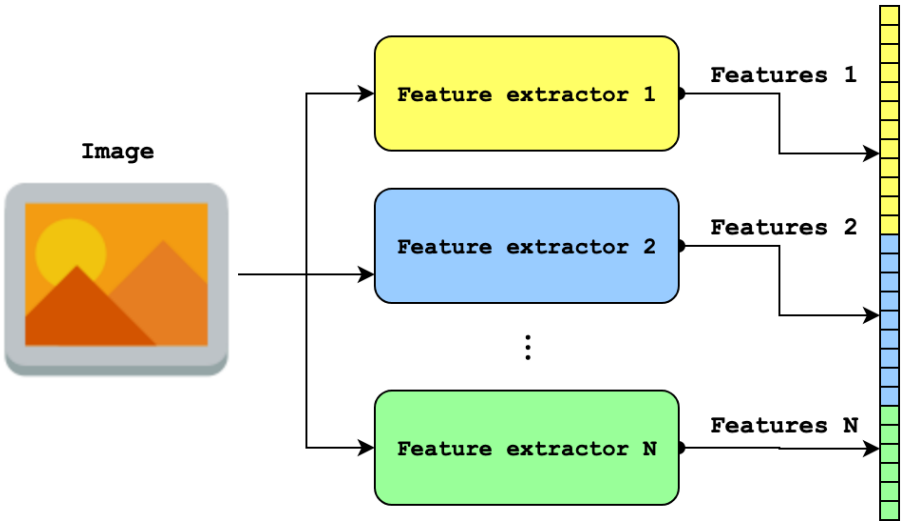


Figure 16 – A schematic diagram illustrating the feature concatenation methodology. Basically, the set of features extracted by a given image extractor is stacked on top of each other and sent to the next method's step, which may be classification. Note that each extractor may produce a different number of features.

the same structure and use it as input to another model, for example, a classifier. This is an effective and simple way to achieve feature fusion, which justifies why this method is widely used.

A slightly different problem is to combine features extracted from images with other features obtained from different sources of data. For example, in medical disease detection, we may have images and patient clinical data available to support the decision over the image (Chai; Liu; Xu, 2018; Liu et al., 2019; Rodrigues; Markou; Pereira, 2019; Pacheco; Krohling, 2020a). In other words, the image is the main source of information and the extra data – usually named as meta-data – provide additional information about the problem. Likewise in the previous problem, a simple concatenation is also the most common approach to combine both images and meta-data (Zhang et al., 2018; Liu et al., 2019). For example, Kharazmi et al. (2018) proposed a feature fusion system based on concatenation and Sparse Autoencoder (SAE) to detect a specific type of skin cancer named Basal Cell Carcinoma; and Sierra and González (2018) also used concatenation to combine image features, extracted using two CNNs architectures, with textual meta-data to perform user gender prediction.

In general, as images are high dimensional data, the number of features extracted from them is much higher than the ones extracted from meta-data. In this context, a simple concatenation may not be effective. Different works have proposed approaches that, first, select/reduce the image features – using Principal Component Analysis (Abdi; Williams, 2010) or a Neural Network, for example - and then perform the concatenation step illustrated in Figure 16 (Audebert; Saux; Lefèvrey, 2017; Viswanath et al., 2017).

This kind of approach assumes that by selecting the image features, the pattern within both set of features may be easier identified by a classifier, for example.

Recently, Li et al. (2020) stated that concatenation approaches are not able to enforce a model to focus on specific parts of the features extracted from images. Particularly, such approach may not consider the potential of the meta-data for visual tasks. Thereby, they proposed the MetaNet, a multiplication-based data fusion approach that uses a sequence of 1D convolution on meta-data to extract coefficients to assist visual features extracted from images in the classification task. The authors applied this method for skin cancer detection and achieved superior performance than a concatenation methodology. However, the method fails to improve melanoma detection, which is the deadliest case of skin cancer. Essentially, there is still room for improvement and better performance may be achieved using a better aggregation methodology.

In this Chapter, we deal with the problem of combining images and meta-data using deep learning models. The key contributions are summarized as follows:

- First, we introduce a baseline method that uses concatenation to combine images and meta-data. This method introduces a straightforward mechanism to control the contribution of each source of data to the combined feature array.
- Next, we propose the Meta-data Processing Block (MetaBlock) a structure that uses meta-data to support data classification by identifying the most relevant features extracted from the images.
- Both methods are designed to deal with Convolutional Neural Networks (CNNs) and are agnostic to the architecture. We apply both of them to classify the ISIC 2019 dataset – a well-known skin lesion dataset composed of images and patient demographics – using six different CNN architectures.
- We compare our results with the CNN models without using meta-data and with the MetaNet, the approach proposed by Li et al. (2020).

3.1 Notation and problem formulation

Let us consider a classification problem in which each sample is composed of an image ($\mathbf{x}_{\text{img}}$), a group of meta-data representing context information ($\mathbf{x}_{\text{meta}}$), and a label $y \in \{1, \cdots, N_{\text{lab}}\}$, where N_{lab} is the number of labels. We represent this problem by the tuple $\{X_{\text{img}}, X_{\text{meta}}, Y\}$. For each type of data X_{img} and X_{meta}, we need to define a feature extractor ψ. For the image, we define a CNN $\psi_{\text{img}} = g_{\text{cnn}}(\mathbf{x}_{\text{img}})$ in which we extract the last feature maps as the image features – see Section 2.3.3. Regarding the meta-data, the extractor ψ_{meta} depends on the type of data, which is a characteristic of the classification

problem. For example, the meta-data is represented by words, we may apply a word embedding method such as Word2Vec (Mikolov et al., 2013), if it is a text file, we may need to apply a more Natural Language Processing (NLP) technique to obtain the features. The only requirement is that ψ_{meta} return features in $\mathbb{R}^{d_{\text{meta}}}$. Therefore, we assume that $\psi_{\text{meta}}(\mathbf{x}_{\text{meta}}) \in \mathbb{R}^{d_{\text{meta}}}$. Lastly, let us define both set of features as:

$$\begin{aligned} \widetilde{\mathbf{x}}_{\text{img}} &= \psi_{\text{img}}(\mathbf{x}_{\text{img}}) \\ \widetilde{\mathbf{x}}_{\text{meta}} &= \psi_{\text{meta}}(\mathbf{x}_{\text{meta}}) \end{aligned} \tag{3.1}$$

where $\widetilde{\mathbf{x}}_{\text{img}} \in \mathbb{R}^{k_{\text{img}} \times m_{\text{img}} \times n_{\text{img}}}$ and $\widetilde{\mathbf{x}}_{\text{meta}} \in \mathbb{R}^{d_{\text{meta}}}$.

At the end, our goal is to propose a method that estimates the probability of y assuming a class $c \in \{1, \cdots, N_{\text{lab}}\}$ given the image and meta-data:

$$\hat{y} = p(y = c \mid \widetilde{\mathbf{x}}_{\text{img}}, \widetilde{\mathbf{x}}_{\text{meta}}) \tag{3.2}$$

These notations are employed for both methods in the next sections.

3.2 Concatenating features based on a contribution factor

In this section, we describe our first approach to combine images and meta-data for a classification problem. As previously described, the most common approach to deal with data combination is the concatenation of features (Audebert; Saux; Lefèvrey, 2017; Viswanath et al., 2017; Pogorelov et al., 2018; Li et al., 2018; Hai et al., 2019). Nonetheless, we can point out two issues in directly applying this approach for the problem described in Section 3.1:

1. The meta-data should be used to support image classification. The most important information is still the image. By just concatenating both sets of features, we are not considering this point.

2. Images are high dimensional data. Even the features extracted from them are usually higher dimensional than meta-data. The result of concatenating a higher dimensional array with a lower one may be ineffective.

Based on these two issues, we proposed a baseline approach that takes advantage of the CNN architecture by including a single-layer neural network to play the role of a feature learning block. In addition, we introduce a straightforward mechanism to control the influence of each set of features generated by each extractor ψ.

Considering a set of feature maps $\widetilde{\mathbf{x}}_{\text{img}}$, we first apply the flatten operation to transform it in an $\widetilde{\mathbf{x}}_{\text{img}} \in \mathbb{R}^{d_{\text{img}}}$ dimensional array – this operation is described in Section

2.3.3. As $\widetilde{\mathbf{x}}_{\text{img}}$ is high dimensional, we define a single-layer neural network to select features from it:

$$\widetilde{\mathbf{s}}_{\text{img}} = \varphi\left(W_s^T \widetilde{\mathbf{x}}_{\text{img}} + \mathbf{w}_{0_s}\right) \tag{3.3}$$

where the matrix of weights $W_s \in \mathbb{R}^{d_{\text{img}} \times h}$ and the bias $\mathbf{w}_{0_s} \in \mathbb{R}^h$. The dimensionality of $\widetilde{\mathbf{s}}_{\text{img}}$ is conditioned to the number of output neurons in the layer, which is represented by h. In other words, h is the number of selected features from $\widetilde{\mathbf{x}}_{\text{img}}$.

In order to determine the number selected features, we propose a combination factor (c_f), which is a mechanism to control and balance the number of image and meta-data features. We define the total number of feature that we want to concatenate as follows:

$$t_{\text{feat}} = \lceil c_f h + (1 - c_f) d_{\text{meta}} \rceil \tag{3.4}$$

where $0 \leq c_f < 1$ and $\lceil\rceil$ is the ceil operator. As we can note from equation 3.4, the combination factor c_f controls the dimensionality of both image and meta-data features. In order to balance the features, we may condition the total number of features according to the d_{meta}:

$$t_{\text{feat}} = \frac{d_{\text{meta}}}{(1 - c_f)} \tag{3.5}$$

Next, conditioned by d_{meta}, we compute h:

$$h = \lceil t_{\text{feat}} - d_{\text{meta}} \rceil \tag{3.6}$$

In brief, the combination factor controls the number of features selected from $\widetilde{\mathbf{x}}_{\text{img}}$. For example, if we have a problem in which $d_{\text{img}} = 1000$, $d_{\text{meta}} = 50$, and $c_f = 0.75$, the number selected features is equal to 150.

Once we determine the array of selected features $\widetilde{\mathbf{s}}_{\text{img}}$, we concatenate it with the meta-data features $\widetilde{\mathbf{x}}_{\text{meta}}$:

$$\widetilde{\mathbf{x}} = \widetilde{\mathbf{s}}_{\text{img}} \oplus \widetilde{\mathbf{x}}_{\text{meta}} \tag{3.7}$$

where $\oplus$ is the concatenation operator. Now, the concatenated array $\widetilde{\mathbf{x}}$ becomes the input to the CNN's fully-connected layer, which returns the $p(y = c \mid \widetilde{\mathbf{x}}_{\text{img}}, \widetilde{\mathbf{x}}_{\text{meta}})$. The full method is illustrated in Figure 17.

As previously described, the feature learning block, defined in 3.3, is a single-layer neural network. Since the CNN is trained by using an end-to-end backpropagation algorithm,

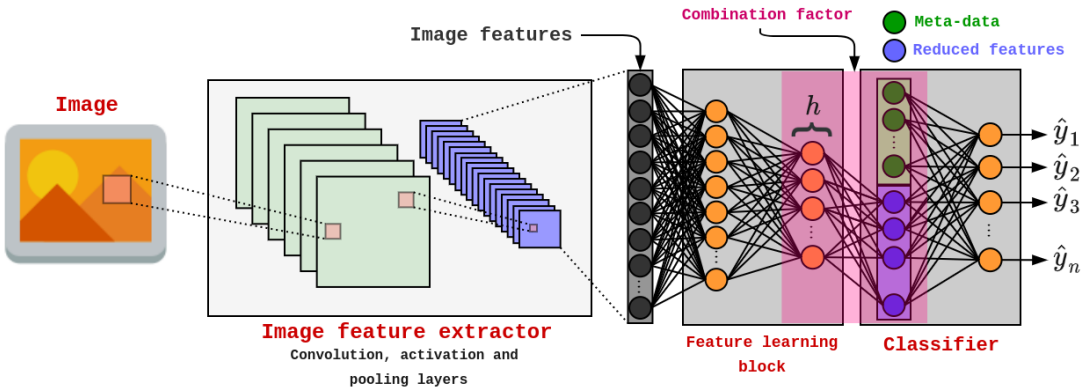


Figure 17 – A diagram illustrating the proposed method to combine image and meta-data features. First, the image features are extracted by the CNN feature extractor. Next, these features are selected according to h. Finally, the features are combined and sent to the models' classifier.

we include the feature learning weights within the CNN's training parameters Θ and use the same loss function $\mathcal{L}(\Theta)$ – see section 2.2.2. Thereby, considering $\Theta_s = \{W_s, \mathbf{w}_{0_s}\}$, we may compute $\nabla_{\Theta_s}\mathcal{L}(\Theta)$ as:

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta_s} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_{\text{fc}}} \frac{\partial \mathbf{u}_{\text{fc}}}{\partial \Theta_{\text{fc}}} \frac{\partial \mathbf{u}_{\text{fc}}}{\partial \mathbf{a}_{\text{s}}} \frac{\partial \mathbf{a}_{\text{a}}}{\partial \Theta_{\text{s}}} \tag{3.8}$$

where $\mathbf{a}_{\text{fc}}$, $\mathbf{u}_{\text{fc}}$, $\mathbf{a}_{\text{s}}$, and $\mathbf{u}_{\text{s}}$ are the activation and pre-activation functions of the fully-connected layer and feature learning block, respectively. The remaining part of the training is carried out according to Algorithm 1. Therefore, the proposed model is trained by an end-to-end backpropagation. This is the main reason we employed a single-layer neural network as the feature learning block. Another option could be applying a traditional feature reducer such as PCA (Abdi; Williams, 2010). However, beyond the fact it is linear, the proposed approach is faster and simpler since the backpropagation is already used to train the image feature extractor layer. Since the computational cost to train CNNs is high, designing approaches that take advantage of its training process is very desired.

3.3 Meta-data Processing Block (MetaBlock): an attention-based mechanism to combine multi-source features

In the previous section, we assume that the contribution of each source can be controlled by determining the number of its features that are sent to the classifier. Although we show in the experiments it works properly, it is actually a strong assumption since a model may learn to assign bigger or smaller weights to the meta-data feature connection regardless of its dimensional size. In addition, the combination factor (c_f) is a parameter that must be defined beforehand.

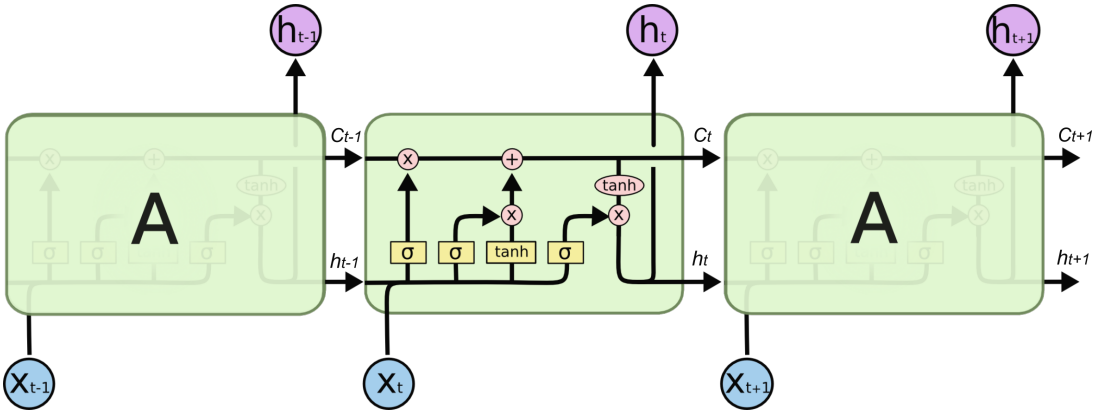


Figure 18 – A schematic diagram of a standard LSTM block. Basically, the block is composed of operations and gates that control and manipulate the information flow. Figure from (Olah, 2015).

In this section, we present the Metadata Processing Block (MetaBlock) an attention-based mechanism approach (Larochelle; Hinton, 2010; Bahdanau; Cho; Bengio, 2014) that uses metadata to enhance the feature maps extracted from images. The proposed approach does not rely on concatenation to combine features and its building blocks are inspired by the Long Short-Term Memory (LSTM) (Hochreiter; Schmidhuber, 1997) gates, which are basic composed of a single layer neural network, an activation function, and an element-wise operation. Both the Attention Mechanism and the LSTM have several variants and it is beyond the scope of this thesis describing them in detail. However, before describing the MetaBlock, we offer a review of the main ideas of both of them that inspired us to create our combination approach.

3.3.1 Long Short-Term Memory (LSTM)

LSTM is a special type of Recurrent Neural Networks (RNN) that is able to learn long-term dependencies over input data (Hochreiter; Schmidhuber, 1997). Consequently, it is widely applied to Natural Language Processing (NLP) problems (Young et al., 2018). As any RNN, it is composed of a chain of repeating blocks that process the input data using a sequence of operations and gates. In Figure 18 the traditional LSTM model is illustrated. The model's input and output are represented by x_t and h_t, respectively. The variable C_t represent the state of each block. The gates (yellow squares) are represented by a single-layer neural network using sigmoid and the tangent hyperbolic as activation functions. The operations (pink circles) are pointwise addition and multiplication between two arrays of data. The basic idea is to select information through the gates and compose them with the addition and multiplication operations (Staudemeyer; Morris, 2019).

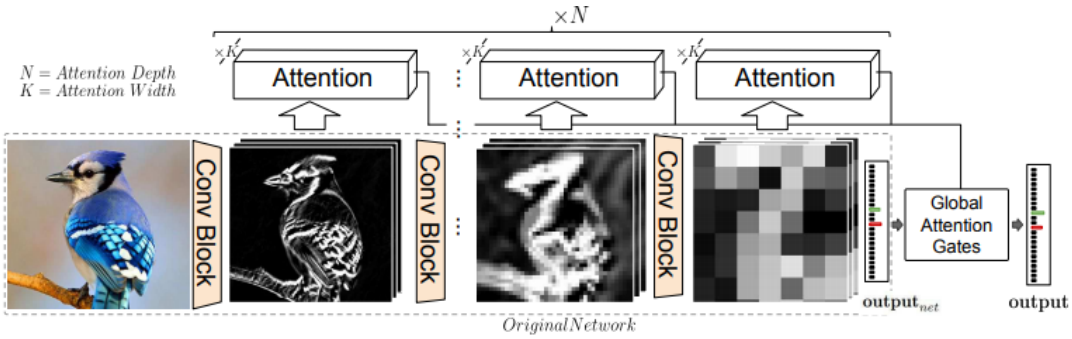


Figure 19 – An attention mechanism proposed by Rodríguez et al. (2018). The original CNN architecture is extended by including several attention blocks. The original model's output is controlled by the global attention gates. Figure from (Rodríguez et al., 2018).

3.3.2 Attention mechanism

Inspired by biological observations on human vision, Larochelle and Hinton (2010) introduced the idea of *fixation point strategy*, which is an attempt to allow deep learning models to concentrate on few relevant features. Later, Bahdanau, Cho and Bengio (2014) proposed the idea of attention in NLP for machine translation. Basically, we can define the attention mechanism as a part of a deep model architecture that allow the own model to highlight relevant features extracted from the input data (Galassi; Lippi; Torroni, 2019). Nowadays, the concept of attention mechanism have been achieving state-of-the-art performance on NLP (Devlin et al., 2018; Radford et al., 2019). It has been also applied to improve interpretability in feature maps extracted using CNNs (Vaswani et al., 2017; Li et al., 2019b; Chaplot et al., 2019).

The way the attention mechanism is included in the model depends on the type of the model and the task it is employed. For example, in the original method, Bahdanau, Cho and Bengio (2014) proposed a single-layer neural network to learn weights from LSTM[1] outputs – the variable h_t in Figure 18 – and use them to scale the cell states (C_t). For image processing with CNNs, the standard approach consist of receiving a feature map as input and apply a function to learn new concepts from this map (Vaswani et al., 2017; Li et al., 2019b). For most of the applications, this function is another neural network mainly because it is easy to train them along with CNN's kernels. Lastly, the output of this function is applied to control another set of features or the CNNs output. In Figure 19 is depicted an attention structure proposed by Rodríguez et al. (2018). As we can note, the authors include several attention blocks that contribute to the global attention gates, which control the model's output.

[1] Actually, they applied a more sophisticated model based on LSTM. For simplicity, we may consider an LSTM.

3.3.3 Methodology

As stated earlier, the Meta-data Processing Block (MetaBlock) algorithm aims to enhance the image according to additional information. In other words, its goal is to enhance the feature maps $\widetilde{\mathbf{x}}_{\text{img}}$ according to the meta-data features $\widetilde{\mathbf{x}}_{\text{meta}}$. In Figure 20 is depicted the MetaBlock's concept. Observe that the output feature $\widetilde{\mathbf{x}}$ will have the same shape of $\widetilde{\mathbf{x}}_{\text{img}}$. However, it is modified to select the most relevant features according to the meta-data features $\widetilde{\mathbf{x}}_{\text{meta}}$.

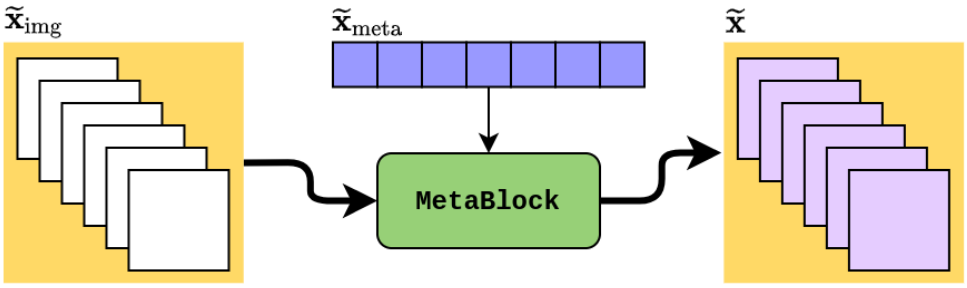


Figure 20 – A schematic diagram illustrating the main idea of the proposed combination approach. The image features are selected according to the meta-data. In the end, the meta-data enhance the image features.

Essentially, we want that the MetaBlock, illustrated in Figure 20, works similarly to an attention mechanism. It must help the model to concentrate on more important features by incorporating the meta-data knowledge into the image feature maps. To achieve this goal, we define the MetaBlock equation similar to the batch normalization one, described in Equation 2.31 – please, refer to Section 2.2.4.3 to refresh the batch normalization concepts. Basically, the idea is to learn a function to scale and shift the image features according to the meta-data and apply LSTM-like gates to select the most relevant features. The MetaBlock equation is defined as follows:

$$\widetilde{\mathbf{x}} = \sigma\left[\tanh\left[f_b(\widetilde{\mathbf{x}}_{\text{meta}}) \times \widetilde{\mathbf{x}}_{\text{img}}\right] + g_b(\widetilde{\mathbf{x}}_{\text{meta}})\right] \tag{3.9}$$

where both $\times$ and $+$ perform an element-wise multiplication and addition, respectively; $\sigma(\cdot)$ and $\tanh(\cdot)$ are the sigmoid and hyperbolic tangent, defined by Equations 2.11 and 2.12, respectively, and are designed to work as the LSTM gates. The block is based on two functions $f_b(\widetilde{\mathbf{x}}_{\text{meta}})$ and $g_b(\widetilde{\mathbf{x}}_{\text{meta}})$. Basically, these functions compose an affine transformation on the image features – similar to the batch normalization idea – which is an effective way to modify features while preserving their dimension. They can be designed as any learnable function. Nonetheless, likewise a LSTM gate, we model both functions as a single-layer neural network since it is a straightforward way to attach a non-linear

function into a deep learning pipeline. Thereby, the $f_b(\widetilde{\mathbf{x}}_{\text{meta}})$ and $g_b(\widetilde{\mathbf{x}}_{\text{meta}})$ are defined as:

$$f_b(\widetilde{\mathbf{x}}_{\text{meta}}) = W_f^T \widetilde{\mathbf{x}}_{\text{meta}} + \mathbf{w}_{0_f} \tag{3.10}$$

$$g_b(\widetilde{\mathbf{x}}_{\text{meta}}) = W_g^T \widetilde{\mathbf{x}}_{\text{meta}} + \mathbf{w}_{0_g} \tag{3.11}$$

where both matrices of weights $\{W_f, W_g\} \in \mathbb{R}^{d_{\text{meta}} \times k_{\text{img}}}$ and the bias $\{\mathbf{w}_{0_f}, \mathbf{w}_{0_g}\} \in \mathbb{R}^{k_{\text{img}}}$ – recall that k_{img} is the number of feature maps extracted from the image. Therefore, each function return k_{img} coefficients, which we name modifiers. The modifier coefficients may be interpreted as weights that modify feature maps in order to enhance them and to help the model concentrating on more important features.

After modifying the feature maps using the modifier coefficients, we select the most relevant features using the hyperbolic tangent and sigmoid functions gates. In other words, these gates decide which features should be let through and which one should not. From Equation 3.9, we describe both gates as follows:

- **hyperbolic tangent gate**: it is the first gate of the MetaBlock and is defined by this part of the Equation 3.9:

$$T_{\text{gate}} = \tanh\left[f_b(\widetilde{\mathbf{x}}_{\text{meta}}) \times \widetilde{\mathbf{x}}_{\text{img}}\right] \tag{3.12}$$

  This gate controls the value of the scaling operator by adding a non-linearity that outputs values ranging from -1 to 1. Essentially, the rationale behind this gate is to increase or reduce the relevance of each feature by modifying its value to the range [-1,1], where 1 is the most relevant and -1 the opposite. The modifier coefficient responsible by the scaling aims to identify this pattern.

- **sigmoid gate**: it is the second gate of the MetaBlock and operates using the output of the previous gate T_{gate} as follows:

$$S_{\text{gate}} = \sigma\left[T_{\text{gate}} + g_b(\widetilde{\mathbf{x}}_{\text{meta}})\right] \tag{3.13}$$

  This gate shifts the values of the previous gate and outputs values ranging from 0 to 1. In other words, it works similarly to the previous one, however, this gate has the power to turn-off a given feature by setting it to zero. Therefore, the key idea is to output the most relevant features.

In Figure 21, the MetaBlock structure is illustrated – to standardize, we follow the same pallet of colors presented in the LSTM block in Figure 18. In brief, we observe that the meta-data is sent to the g_b and f_b function, which will produce the modifier coefficients.

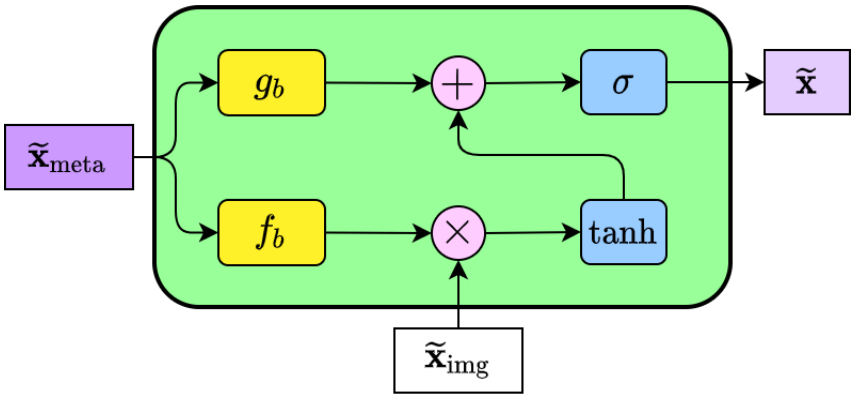


Figure 21 – The internal structure of the Meta-data Processing Block. In summary, the block learns how to modify the image features based on the meta-data features. The output features array has the same shape as the image features.

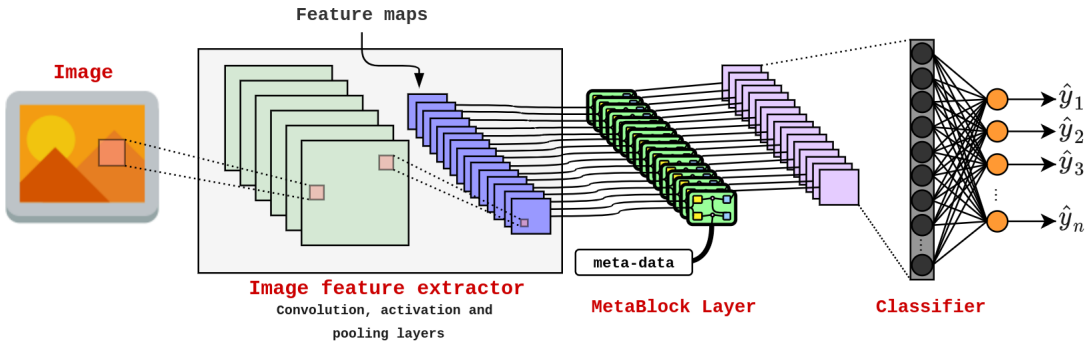


Figure 22 – An illustration of a MetaBlock layer attached to a CNN model. Observe that the meta-data are used to identify the most relevant features from the feature maps before sending them to the classifier.

Next, the feature maps are scaled by the f_b, and the output is modified by the hyperbolic tangent gate. Finally, the result of the previous operation is shifted by the output of the g_b and selected by the sigmoid gate, which produces the block output. An illustrative example of this method is described in the next subsection.

In Figure 22 is illustrated a CNN schematic in which the MetaBlock is used to combine the meta-data and the feature maps. Observe that the MetaBlock outputs are used to feed the classification layer. However, they can be used to feed any type of layer before sending them to the classifier. Also, it is important to note that this method is agnostic to the CNN architecture. As we show in the experiments, it can be easily adapted to different architectures. Finally, in Appendix B.3, we show that, in the worse case scenario, the MetaBlock parameters increase the total number of parameters of a CNN in 0.85%, which is insignificant in terms of training time.

The process of training the functions f_b and g_b is similar to the previous method described in Section 3.2. It is also applied an end-to-end backpropagation. In this case, we have $\Theta_f = \{W_f, \mathbf{w}_{0_f}\}$ and $\Theta_g = \{W_g, \mathbf{w}_{0_g}\}$, which are also included into the CNN's

training parameters Θ. To compute $\nabla_{\Theta_f}\mathcal{L}(\Theta)$ and $\nabla_{\Theta_g}\mathcal{L}(\Theta)$, we can also apply Equation 3.8. Finally, we apply batch normalization – see Section 2.2.4.3 – to regularize the functions f_b and g_b.

3.3.4 Illustrative example

In this section, we present a numerical example to illustrate how the MetaBlock works. To do so, let us consider a CNN model that output 10 feature maps with shape equal to 4×4. For this example, each of the g_b and f_b functions must learn 10 modifiers coefficients to scale and shift each of the 10 feature maps as we describe in the previous section. Let us also consider the meta-data dimension $d_{\text{meta}} = 5$. Thus, the weight matrices and bias, described in Equations 3.10 and 3.11, are defined as $\{W_f, W_g\} \in \mathbb{R}^{5\times10}$ and the bias $\{\mathbf{w}_{0_f}, \mathbf{w}_{0_g}\} \in \mathbb{R}^{10}$. Each set of weights and bias related to g_b and f_b is responsible to learn the scale and shift modifiers, let us say **sc** and **sh**, respectively.

Now, let us consider that g_b and f_b were trained along with the CNN model and learned the following weights and bias:

$$W_f = \begin{bmatrix} -0.408 & 0.107 & -0.039 & 0.816 & 0.260 & -0.332 & -0.983 & 0.125 & 0.277 & 0.034 \\ -0.927 & -0.851 & -0.734 & -0.272 & 0.308 & 0.183 & -0.823 & -0.458 & -0.218 & -0.045 \\ 0.801 & -0.810 & 0.592 & -0.909 & -0.812 & 0.985 & -0.412 & -0.548 & 0.321 & 0.874 \\ 0.422 & -0.982 & -0.737 & -0.796 & -0.144 & 0.378 & -0.974 & 0.927 & -0.368 & -0.526 \\ 0.867 & 0.821 & 0.552 & 0.441 & -0.750 & -0.396 & -0.775 & 0.507 & 0.468 & -0.410 \end{bmatrix} \tag{3.14}$$

$$\mathbf{w}_{0_f} = \begin{bmatrix} 0.906 & 0.799 & 0.953 & -0.262 & 0.492 & 0.26 & 0.518 & -0.402 & -0.271 & 0.694 \end{bmatrix} \tag{3.15}$$

$$W_g = \begin{bmatrix} -0.898 & -0.785 & 0.549 & -0.686 & -0.856 & 0.929 & 0.267 & -0.720 & -0.471 & 0.997 \\ -0.686 & 0.425 & -0.570 & 0.240 & 0.104 & 0.429 & 0.393 & 0.372 & -0.801 & -0.105 \\ 0.571 & 0.716 & -0.419 & -0.565 & 0.592 & -0.541 & -0.835 & -0.465 & -0.788 & -0.603 \\ -0.437 & -0.903 & -0.783 & -0.043 & -0.571 & -0.035 & 0.791 & 0.565 & 0.385 & 0.997 \\ 0.957 & 0.304 & 0.861 & 0.572 & -0.047 & -0.638 & 0.367 & 0.310 & 0.258 & 0.459 \end{bmatrix} \tag{3.16}$$

$$\mathbf{w}_{0_g} = \begin{bmatrix} 0.906 & 0.799 & 0.953 & -0.262 & 0.492 & 0.26 & 0.518 & -0.402 & -0.271 & 0.694 \end{bmatrix} \tag{3.17}$$

Considering the meta-data array as:

$$\mathbf{x}_{\text{meta}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0.7 \end{bmatrix} \tag{3.18}$$

Through Equations 3.10 and 3.11 we compute the values of the scale and shift modifier coefficients, which results in:

$$\mathbf{sc} = \begin{bmatrix} 0.181 & 0.634 & 0.571 & 0.590 & 0.54 & -0.176 & -1.834 & -0.383 & 0.120 & 0.400 \end{bmatrix} \tag{3.19}$$

$$\mathbf{sh} = \begin{bmatrix} -0.008 & 0.651 & 1.534 & -0.307 & -0.292 & 1.171 & 1.434 & -0.533 & -1.362 \end{bmatrix} \tag{3.20}$$

These values are used to enhance each of the 10 feature maps. For example, let us consider the first feature maps as:

$$F_1 = \begin{bmatrix} 0.628 & -0.434 & -0.223 & -0.971 \\ 0.098 & -0.249 & 0.776 & -0.271 \\ 0.846 & 0.818 & -0.321 & 0.270 \\ -0.643 & -0.583 & 0.336 & 0.749 \end{bmatrix} \tag{3.21}$$

we compute the MetaBlock for this feature map as follows:

$$\tilde{F}_1 = \sigma\left[\tanh\left[0.181 \times F_1\right] + (-0.008)\right], \tag{3.22}$$

which results in the following feature map:

$$\tilde{F}_1 = \begin{bmatrix} 0.526 & 0.479 & 0.488 & 0.455 \\ 0.502 & 0.487 & 0.532 & 0.486 \\ 0.535 & 0.534 & 0.484 & 0.510 \\ 0.470 & 0.472 & 0.513 & 0.531 \end{bmatrix} \tag{3.23}$$

Lastly, we just need to repeat this process using the remaining values in **sc** and **sh** to the other 9 feature maps. Obviously, these operations are computed at once through vectorization. It is worth mentioning that all values using in this section are fictitious. They are just for understanding purposes. In a real situation, the feature maps matrices are much bigger.

3.4 Experimental results

In this section, we carry out experiments to evaluate the performance of both combination approaches proposed in this chapter: the baseline using the contribution factor and the Meta-data Processing Block (MetaBlock). We use six different CNN architectures trained on ISIC 2019 dataset (ISIC, 2019), which is composed of dermoscopy skin lesion images and patient demographics. Thereby, we start this section by describing the experiment's setup, including the dataset and the deep models. Next, we present a discussion about the results.

3.4.1 Experiments setup

We evaluate the combination approaches using the following CNN architectures: EfficientNet-B1 and B4 (Tan; Le, 2019), DenseNet-121 (Huang et al., 2017b), MobileNet-v2 (Sandler et al., 2018), ResNet-50 (He et al., 2016), and VGG-13 (Simonyan; Zisserman,

2014). All models are trained on ISIC 2019 dataset (ISIC, 2019), a skin lesion archive containing 25,331 dermoscopy images, patient demographics, eight skin lesions: Melanoma (MEL), Melanocytic nevus (NV), Basal cell carcinoma (BCC), Actinic keratosis (AK), Benign keratosis (BKL), Dermatofibroma (DF), Vascular lesion (VASC), and Squamous cell carcinoma (SCC). We compare the six models with and without using the patient demographics as meta-data along with the images.

In order to combine both types of data, we attach the baseline concatenation method, the MetaBlock, and MetaNet (Li et al., 2020) into the original architecture of the six CNN models employed in this experiment. All models were pre-trained on ImageNet (Deng et al., 2009) and fine-tuned on ISIC for 150 epochs using SGD optimizer with a learning rate equal to 0.001, momentum equal to 0.9, and weight decay equal to 0.001. The learning rate is reduced by a rate of 0.1 if the model does not improve for 10 consecutive epochs. In addition, we used early stopping if the model does not improve for 15 consecutive epochs. As the dataset is imbalanced, i.e., the samples are not represented equally among the labels, we applied the weighted cross-entropy as the loss function in which the weights are determined according to the labels' frequency. All images were resized to 224×224 and we applied data augmentation using common image processing operations: horizontal and vertical flips, adjustments in brightness, contrast, and, saturation, image scaling, and random noise (Perez et al., 2018; Gessert et al., 2019; Pacheco; Ali; Trappenberg, 2019; Kather et al., 2019).

For all experiments, we reserved approximately 10% of each dataset for testing respecting the labels' frequency distribution. The remaining 90% of data were used on the training phase through 5-fold cross-validation for assessing the effectiveness of the models[2]. To measure the performance, we computed the average and standard deviation of the following metrics: accuracy (ACC), balanced accuracy (BACC), the aggregated area under the curve (AUC), and the cross-entropy (Loss). As the dataset is imbalanced, we consider BACC as the main metric.

The patient demographics contain information about age, represented by an integer, gender, and anatomical region of the skin lesion, both of them represented by strings. In order to use this information as meta-data, we need to transform the categorical data into scalar numbers. The attribute of gender may assume two values: `male` or `female`. On the other hand, the anatomical region may assume six different values: `anterior torso`, `head/neck`, `lateral torso`, `lower extremity`, `oral/genital`, `palms/soles`, `posterior torso`, and `upper`. Thereby, we applied the one-hot encoding strategy (Zhang et al., 2020; Géron, 2019) to encode the patient demographics into an array of 11 values – particularly, Li et al. (2020) employed the same strategy to use MetaNet. For example, for the gender

[2] We set this configuration because this experiment will also be used for the ensemble experiments in the next section, which demands training, validation, and test partitions.

attribute, the employed strategy may assume [1,0], [0,1], or [0,0] if gender is `male`, `female`, or if this information is missing, respectively. We applied the one-hot encoding strategy mainly because the data do not demand a fancy strategy to encode it properly, i.e., the values that each attribute assumes are well defined and in a set of possibilities; and the method is simple to implement and allow us to map the missing data. Therefore, the one-hot encode may be seen as the meta-data extractor ψmeta that results in an array in $\mathbf{x}_{\text{meta}} \in \mathbb{R}^{11}$, where $d_{\text{meta}} = 11$.

Lastly, for the baseline concatenation methodology, we use the contribution factor equal to 0.8, i.e., 80% of features coming from the image and 20% from the meta-data. Thus, the number of neurons of the feature selector block h is equal to 44 – see equation 3.6. A sensitivity analysis and a discussion about the combination factor may be found in Appendix B.1. In order to compare the experiments results, we perform the non-parametric Friedman test following by the Wilcoxon test (if applicable), using $p_{\text{value}} = 0.05$ (Derrac et al., 2011). We also perform the A-TOPSIS (Krohling; Pacheco, 2015), a straightforward method to rank algorithms in terms of mean and standard deviation.

3.4.1.1 Experiment results

We now present the results obtained for the ISIC 2019 dataset with and without considering the meta-data. In Tables 1, 2, 4, and 3 are presented the results, in terms of mean and standard deviation, for each CNN model without using meta-data, for the models using the concatenation approach, using the MetaBlock, and for MetaNet, respectively. To ease the visualization, in Table 5 is presented the performance only in terms of BACC for each method.

No meta-data				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B1	0.746 ± 0.029	0.735 ± 0.01	0.952 ± 0.003	0.724 ± 0.053
EfficientNet-B4	0.776 ± 0.023	0.755 ± 0.013	0.956 ± 0.004	0.654 ± 0.051
DenseNet-121	0.774 ± 0.029	0.755 ± 0.024	0.961 ± 0.006	0.645 ± 0.062
MobileNet-v2	0.763 ± 0.008	0.742 ± 0.009	0.961 ± 0.001	0.647 ± 0.007
ResNet-50	0.767 ± 0.036	0.744 ± 0.035	0.959 ± 0.007	0.667 ± 0.076
VGGNet-13	0.722 ± 0.018	0.701 ± 0.016	0.948 ± 0.004	0.761 ± 0.041

Table 1 – Performance of the CNN models considering only the dermoscopy images.

As we can note from the tables, the use of the patient demographics provides a visible performance improvement in terms of BACC and Loss, in particular when the MetaBlock is applied. As the dataset is imbalanced, the ACC is slightly higher than the BACC for all models, which is expected. To compare the four methods, we performed the Friedman and Wilcoxon tests, considering the BACC metric as the tests input. The Friedman test returned $p_{value} = 0.0011$. Thus we performed the pairwise Wilcoxon test, which is described in Table 6. As we can see, the test indicates that the MetaBlock approach

Concatenation				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B1	0.735 ± 0.01	0.729 ± 0.006	0.949 ± 0.001	0.731 ± 0.013
EfficientNet-B4	0.784 ± 0.004	0.768 ± 0.015	0.96 ± 0.002	0.662 ± 0.03
DenseNet-121	0.738 ± 0.012	0.737 ± 0.01	0.952 ± 0.002	0.714 ± 0.023
MobileNet-v2	0.716 ± 0.009	0.723 ± 0.016	0.946 ± 0.004	0.759 ± 0.03
ResNet-50	0.729 ± 0.013	0.726 ± 0.013	0.948 ± 0.005	0.745 ± 0.029
VGGNet-13	0.724 ± 0.015	0.729 ± 0.013	0.949 ± 0.004	0.743 ± 0.036

Table 2 – Performance of the CNN models using the baseline concatenation approach to combine the dermoscopy images with the patient demographics.

MetaBlock				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B1	0.734 ± 0.027	0.731 ± 0.03	0.951 ± 0.008	0.736 ± 0.075
EfficientNet-B4	0.807 ± 0.008	0.762 ± 0.011	0.962 ± 0.003	0.581 ± 0.025
DenseNet-121	0.800 ± 0.011	0.769 ± 0.013	0.965 ± 0.002	0.577 ± 0.026
MobileNet-v2	0.777 ± 0.006	0.76 ± 0.004	0.958 ± 0.001	0.631 ± 0.017
ResNet-50	0.804 ± 0.006	0.771 ± 0.011	0.966 ± 0.002	0.569 ± 0.015
VGGNet-13	0.753 ± 0.031	0.74 ± 0.015	0.955 ± 0.007	0.683 ± 0.067

Table 3 – Performance of the CNN models using the MetaBlock to combine the dermoscopy images with the patient demographics.

MetaNet				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B1	0.742 ± 0.029	0.743 ± 0.018	0.953 ± 0.007	0.729 ± 0.085
EfficientNet-B4	0.766 ± 0.019	0.756 ± 0.012	0.959 ± 0.004	0.681 ± 0.025
DenseNet-121	0.725 ± 0.020	0.723 ± 0.016	0.949 ± 0.005	0.756 ± 0.051
MobileNet-v2	0.742 ± 0.019	0.731 ± 0.015	0.955 ± 0.006	0.713 ± 0.052
ResNet-50	0.753 ± 0.030	0.746 ± 0.022	0.956 ± 0.008	0.696 ± 0.081
VGGNet-13	0.767 ± 0.020	0.746 ± 0.013	0.959 ± 0.005	0.696 ± 0.066

Table 4 – Performance of the CNN models using the MetaNet to combine the dermoscopy images with the patient demographics.

is statistically different from the remaining ones. We also performed the A-TOPSIS to rank the three methods. The rank is presented in Figure 23 and it is in accordance with the statistical test, i.e., the MetaBlock approach has a higher rank score and the three remaining methods achieve quite similar values.

To get a gist of the AUC metric, in Figure 24 is shown the ROC curves, considering the ResNet-50 model, with no meta-data, concatenation, MetaBlock, and MetaNet approaches considering macro average and melanoma, the deadliest type of skin cancer. As we can see, both ROC curves for the MetaBlock approach are above the remaining methods, which is in line with the previous results. In Figure 25 is shown the confusion matrix, also considering the ResNet-50[3] model, for each methodology. As seen in the

[3] As we would have 18 confusion matrices and 18 ROC curves, we decided to present these plots only for the ResNet-50 because it achieves a proper average performance without using the meta-data.

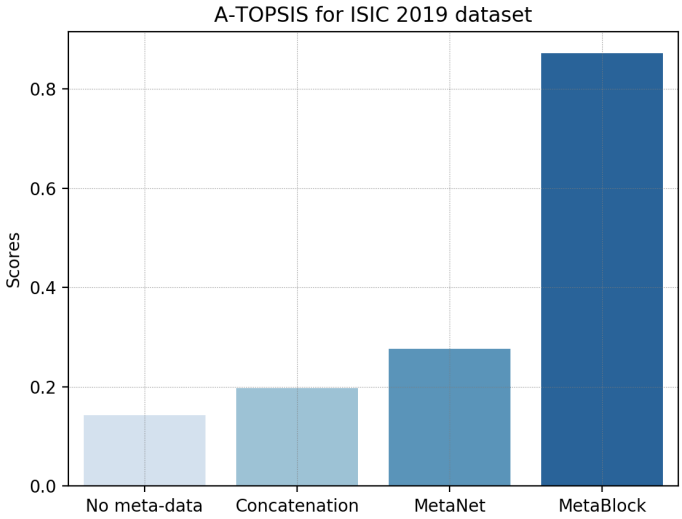
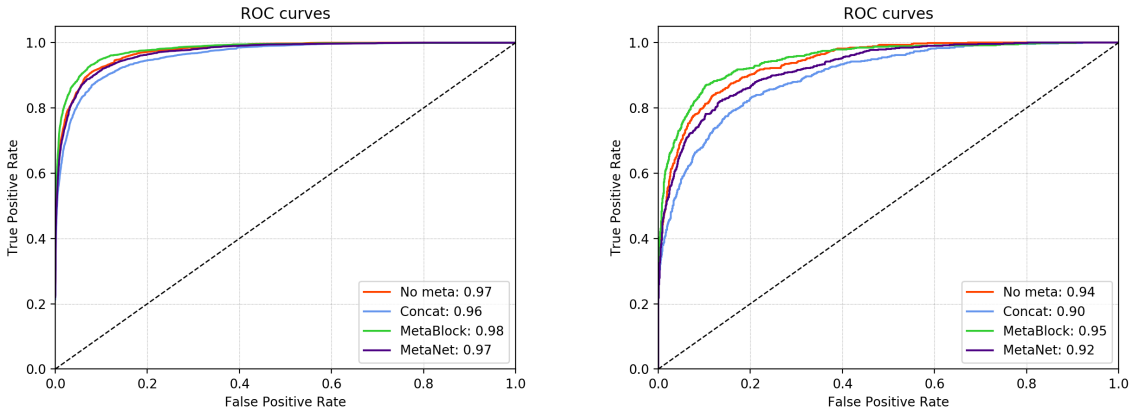


Figure 23 – The A-TOPSIS rank for the four methods considering the BACC metric.



(a) Macro average ROC (b) Melanoma ROC

Figure 24 – The Macro average and melanoma ROC curves for no meta-data, concatenation, MetaBlock, and MetaNet approaches considering the ResNet-50 model.

Comparing all approaches				
Model	**No meta-data**	**Concatenation**	**MetaBlock**	**MetaNet**
EfficientNet-B1	0.735 ± 0.01	0.729 ± 0.006	0.731 ± 0.03	$\mathbf{0.743 \pm 0.018}$
EfficientNet-B4	0.755 ± 0.013	$\mathbf{0.768 \pm 0.015}$	0.762 ± 0.011	0.756 ± 0.012
DenseNet-121	0.755 ± 0.024	0.737 ± 0.01	$\mathbf{0.769 \pm 0.013}$	0.723 ± 0.016
MobileNet-v2	0.742 ± 0.009	0.723 ± 0.016	$\mathbf{0.76 \pm 0.004}$	0.731 ± 0.015
ResNet-50	0.744 ± 0.035	0.726 ± 0.013	$\mathbf{0.771 \pm 0.011}$	0.746 ± 0.022
VGGNet-13	0.701 ± 0.016	0.729 ± 0.013	0.74 ± 0.015	$\mathbf{0.745 \pm 0.013}$

Table 5 – Comparing the performance of the all methodologies in terms of BACC. In bold is highlighted the highest average for each model.

Pair	$p_{\mathbf{value}}$
No meta-data - Concatenation	0.502
No meta-data - MetaBlock	0.002
No meta-data - MetaNet	0.773
MetaBlock - Concatenation	0.001
MetaBlock - MetaNet	0.017
MetaNet - Concatenation	0.343

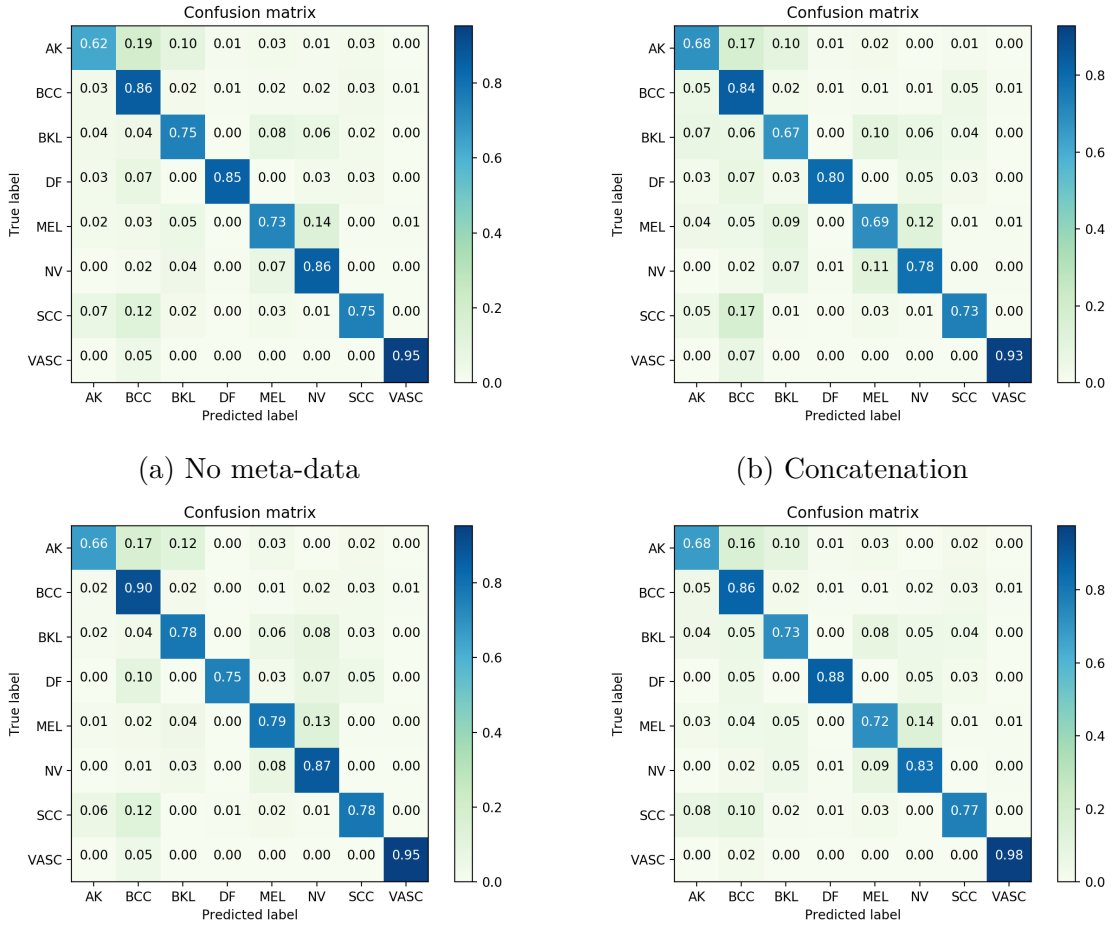
Table 6 – The result of the Wilcoxon pairwise test for all methods.

main diagonal of the confusion matrices, the MetaBlock approach improves almost all true positive values, which reflects in the ROC curves and in the BACC metric. On the other hand, the concatenation and MetaNet approaches reduce detection, in particular, for melanoma.

3.4.1.2 Discussion

The results presented in the previous section indicate that combining meta-data and patient demographics may improve the performance of CNN models for skin cancer detection. However, the results also show the improvement depend on the combination method. As indicated by the statistical test, the concatenation and MetaNet approaches do not present differences from the models without using the meta-data. It happens because these methods present better performances than the no meta-data one only for some models – for example, MetaNet presents higher performance only for EfficientNet-B1 and VGGNet-13. On the other hand, MetaBlock is more stable and presents an average improvement of over 1% in balanced accuracy and reduced the loss for all methods, except the EfficientNet-B1. By reducing the loss function, the method indicates it can predict the skin lesion diagnostic with more certainty, which is very desired for the skin cancer detection task.

Analyzing the ROC curves for the ResNet-50 depicted in Figure 24, the concatenation and MetaNet approaches present a fairly lower performance for both macro average and melanoma curves. In the case of melanoma detection, the confusion matrices in Figure



(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 25 – The confusion matrix for each methodology considering the ResNet-50 model.

25 are also in line with the ROC curves. We believe this lower performance happens because the ISIC 2019 dataset has only three meta-data attributes, age, anatomical region, and gender, and the concatenation approach cannot handle it properly. Conversely, the MetaBlock approach could handle this issue well and presented a higher performance compared to all other approaches. In general, the experiment carried out in this section shows that the MetaBlock approach is a promising tool for combining dermoscopy images and patient demographics, even when just a few information is available. We will return to discuss this topic in Chapter 5, where we present a thorough case study that contains much more meta-data to be used. Also, in Appendix B.2 we assess the contribution of each f_b and g_b in the MetaBlock.

To conclude, Li et al. (2020) analyzed the MetaNet only in terms of recall over the labels, which also shows poor performance for melanoma detection. In addition, the authors do not provide any statistical test or another method to further compare their method with the no meta-data and concatenation approaches. In this experiment, we show that MetaNet's performance depends on CNN architecture and, most of the time, it is similar to the concatenation approach.

4 Learning dynamic weights for an ensemble of deep models

In order to learn a distribution that represents a set of data, Machine Learning models must make assumptions about the data. In theory, these models are able to generalize well using a finite set of samples. However, these samples do not represent all possible data generated by the true distribution. In this context, data-driven algorithms are able to learn rules that may be correct for most samples, i.e., there is a probability that this rule is correct. The *No Free Lunch* theorem Wolpert (1996) states that there is no universal Machine Learning algorithm that provides the best performance for all problems. The only way to know which model is the best one for a task is evaluating it for that task (Géron, 2019). However, it is impracticable to test all possible models. Thus, an effective approach to deal with this problem is trusting in an ensemble of models instead of only one.

Multiple Classifier Systems (MCSs)[1] is the subfield of Machine Learning that proposes algorithms to combine decisions from a group classifiers in order to obtain a more accurate solution. Basically, an MCS is composed of three phases (Britto Jr; Sabourin; Oliveira, 2014; Cruz; Sabourin; Cavalcanti, 2018):

1. **Generation**: this phase aims to create a set of base classifiers. These classifiers should present some diversity in order to improve the group's generalization. There are several strategies to enforce diversity into a group of classifiers, such as using different models or architectures, initializing the models using different methodologies, using different hyperparameters, training the models for different datasets, among others. These strategies can be applied alone or combining each other. When the pool of classifiers is created using different models or architectures it is called a heterogeneous ensemble. Conversely, when all classifiers are instances of the same model it is named a homogeneous ensemble.

2. **Selection or Pruning**: in this phase, one or a set of classifiers is selected from the pool according to a static or dynamic methodology. In brief, the static method selects classifiers using the same rule for all unknown samples presented to the pool. For example, we could select the classifiers according to their accuracy in the validation partition. Conversely, the dynamic method selects classifiers according to the unknown sample that the model is classifying.

[1] Like most of the authors, we use the terms MCS, ensemble of classifiers, and committee of classifiers interchangeably

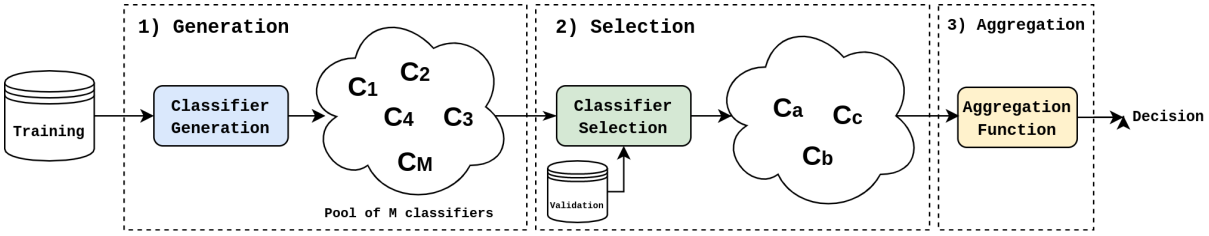


Figure 26 – The workflow of the phases of an MCS. In the first step, a pool of classifiers is created. Next, the selection phase is performed using the validation partition. Finally, the classifiers are aggregated. Adapted from (Cruz; Sabourin; Cavalcanti, 2018).

3. **Aggregation or Fusion**: this phase consists of the aggregation of outputs obtained by the selected classifiers according to some function. This function may be non-trainable – for example, majority voting schemes, average or maximum prediction –, trainable – such as the Mixture of Experts (Masoudnia; Ebrahimpour, 2014) in which the aggregation function is trained along with the classifiers –, or dynamic weights, when the function weights the classifiers for each unknown sample on the fly.

These three phases are illustrated in Figure 26. It is important to note that this representation is not rigid. For example, we can use MCS without a selection phase. In addition, the selection and aggregation phases definitions may overlap, which is the case when a dynamic weighting function is applied.

Three pioneers methods, but still widely used, to build an MCS pipeline are known as boosting (Schapire et al., 1998), bagging (Breiman, 1996), and stacking (Wolpert, 1992). Boosting is a homogeneous method that creates a chain of models that are sequentially trained in a adaptive way. Each model in this sequence tries to correct the predictions error of prior models in the chain. Bagging, short for bootstrap aggregating, is also a homogeneous method, however, the models are trained in parallel using different subsets that are randomly drawn, with replacement, from the entire training data (Polikar, 2006). Stacking is quite similar to bagging, however, this method achieves diversity by using a heterogeneous ensemble. In both cases, we may or may not include a selection phase and the ensemble's prediction is obtained using an aggregation function, as shown in Figure 27.

In Deep Learning, similar to Machine Learning, the performance of models may vary due to multiple reasons such as weights' initialization, hyperparameters, data variance, and overfitting. An ensemble of deep models is a common and effective strategy to deal with this issue (Codella et al., 2017b; Harangi, 2018; Valle et al., 2020; Qummar et al., 2019), since it is known to be more robust and accurate than single deep models (Huang et al., 2017a). Also, in Deep Learning, it is more common to use stacking or bagging approaches since the computational burden of boosting makes it impracticable to handle deep models.

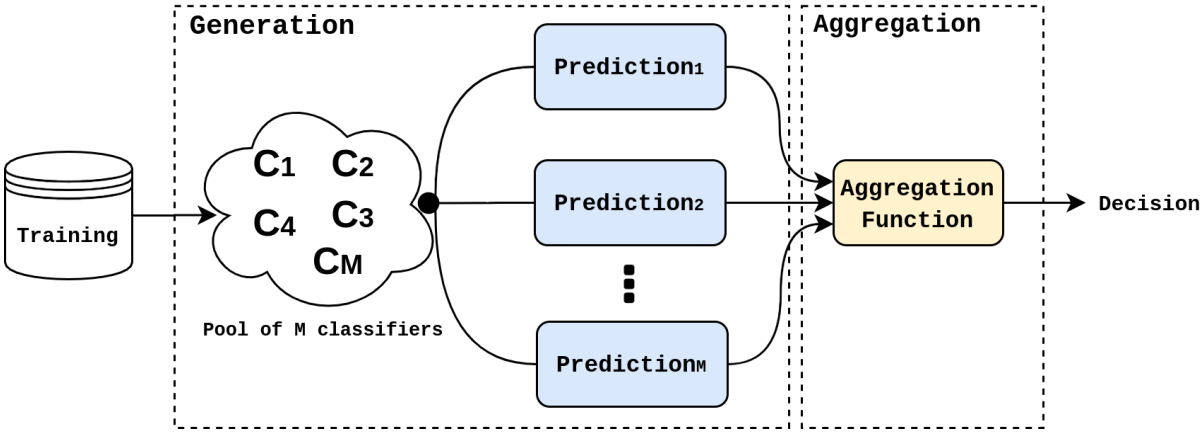


Figure 27 – An illustration of an MCS based on stacking without applying the selection phase. The ensemble decision is obtained through the aggregation of all classifiers predictions.

Currently, this approach is applied by the top-ranked deep learning models in state-of-the-art computer vision challenges such as ImageNet (Deng et al., 2009), International Skin Imaging Collaboration (ISIC, 2019), and Chest X-Ray Competition (Irvin et al., 2019).

Usually, MCS pipelines for Deep Learning models do not include the selection phase (Codella et al., 2017b; Harangi, 2018; Qummar et al., 2019). After the generation phase, the model's predictions are aggregated, typically, using common aggregation operators such as the majority voting (Xiao et al., 2018; Wu et al., 2019b), predictions average (Codella et al., 2017b; Perez; Avila; Valle, 2019; Gessert et al., 2019; Pacheco; Ali; Trappenberg, 2019), maximum prediction (Harangi, 2018), and product of predictions (Harangi, 2018). Other known information fusion techniques such as Choquet integral (Štefka; Holeňa, 2015; Pacheco; Krohling, 2018a) and Dempster-Shafer theory (Quost; Masson; Denœux, 2011) are avoided due to the computational burden, which is an important issue since deep learning itself is computationally expensive. Obviously, a drawback of skipping the selection phase is that all models have the same importance in the ensemble. In other words, it is not possible to identify weak models that may negatively impact on the ensemble performance.

As previously stated, the selection phase may be clustered into static or dynamic methods. The most common way to statically select classifiers from an ensemble is through some classification metric, for example, we may select or weight classifiers predictions according to their classification accuracy on validation partition (Kuncheva, 2014; Nguyen et al., 2020). Search algorithms such as greedy search (Partalas; Tsoumakas; Vlahavas, 2008) and evolutionary algorithms (Santos; Sabourin, 2011; Nguyen et al., 2014) are also common, however, they are time consuming. In general, static methods are unable to online select/weight each unknown sample that is evaluated by the ensemble. This drawback is the motivation for the rise of dynamic selection methods, which basically aim to identify strong classifiers for each unknown sample. Given its relevance, Dynamic Selection (DS)

has become a subfield of MCS.

Over the past few years, several DS algorithms have been proposed to deal with MCS composed of general Machine Learning models (Britto Jr; Sabourin; Oliveira, 2014). Before reviewing DS algorithms, it is important to clarify a key concept regarding DS and dynamic weighting. A loose definition will consider both methods as the same, however, they are slightly different. A DS algorithm aims to estimate the competence of the base classifiers taking into account the local region of the feature space where the unknown sample is located. In other words, it selects classifiers based on feature spaces and usually demands some adaptation on the classifier or another method, such as K-nearest neighbors (KNN) or clustering techniques, to identify the local region (Cruz; Sabourin; Cavalcanti, 2018). On the other hand, a dynamic weighting algorithm also aims to estimate the competence of the base classifiers on the fly, however, the competence is measured as weights/scores that usually are based on the classifier's predictions. These weights may be used to select/prune the ensemble of classifiers, however, the main goal of this approach is to provide weights for a given aggregation function. This is the reason that most authors include dynamic weighting within the aggregation phase instead of the selection one.

DS algorithms have been applied to an ensemble of general machine learning models since the 90's. To name a few, Sabourin et al. (1993) proposed the Classifier Rank (DCS-Rank) which is basically an algorithm that rewards classifiers that correctly classifies the highest number of consecutive samples. Later, Woods, Kegelmeyer and Bowyer (1997) introduced the Overall Local Accuracy (OLA) in which the level of competence of a base classifier is computed according to the classification accuracy in the region of competence. Both methods use the KNN to identify the region of competence and select only one classifier from the pool. Cruz et al. (2015) proposed the Meta-learning Dynamic Ensemble (META-DES), a framework to select base classifiers that combines KNN and meta-learning to identify the region of competence and select the models, respectively. Brun et al. (2016) developed the Dynamic Selection On Complexity (DSOC) which is a new method that takes into account the local classifiers' performance and the data complexity measure (Ho; Basu, 2002) of the test and training data. Lastly, Oliveira, Cavalcanti and Sabourin (2017) introduced the Frienemy Indecision Region Dynamic Ensemble Selection (FIRE-DES) which is a framework for two-class problems that pre-selects base classifiers according to an indecision region in the region of competence. In general, all these methods are commonly applied to a pool of classifiers composed of traditional Machine Learning algorithms, such as Support Vector Machines, Decision Tree, and Naive Bayes (Fatima; Pasha et al., 2017). For more complex models, such as the Deep Learning ones, DS methods are not common because it is harder to find a competence region and the computational burden of these types of models. Thereby, dynamic weighting appears as a feasible approach to tackle these issues.

Dynamic weighting is also widely applied to MCS. One of the first methods, introduced by Merz (1999), is the Stacking Correspondent Analysis Nearest Neighbor (SCANN). As the name suggests, this method generates an ensemble using the stacking approach, performs a Correspondent Analysis (CA) (Greenacre, 1984) using the matrix of predictions and a KNN to compute the weights. As the CA is performed to the matrix containing all predictions, this method is problematic to large datasets. Cevikalp and Polikar (2008) proposed the Local Classifier Weighing Quadratic Programming (LCS-WP) a method that combines local classifier accuracies and quadratic optimization to determine the best weights of the most competent base classifiers. Lastly, Krawczyk and Woźniak (2016) proposed a straightforward approach based on the Gaussian function and the classifier's predictions to estimate the weights of each model within the ensemble. Although simple, it depends on two parameters that may be tricky to set.

In this chapter, we describe our proposed approach to learn dynamic weights for an ensemble of classifiers based on the Dirichlet distribution (Ng; Tian; Tang, 2011) and Mahalanobis distance (Maesschalck; Jouan-Rimbaud; Massart, 2000). This approach presents three main advantages: 1) it is based on the models' predictions which makes it agnostic to the model and can be easily applied to deep learning models; most of the method's computation is done offline using the validation partition, which makes the ensembles' inference fairly fast; 3) the method does not have hyperparameters to set. The key contributions of this chapter are summarized as follows:

- We propose a new approach to learn dynamic weight for an ensemble of deep models. In brief, we estimate the Dirichlet distribution of the models' predictions and apply the Mahalanobis distance to determine the weight of each model.
- The weights learned by the proposed approach are used to reduce the impact of weak models' predictions on the aggregation operator.
- We show that it is possible to use the learned weights to select the most competent models before the aggregation operator.
- We apply the method using six well-known deep learning models to four different medical imaging datasets. We present a discussion to demonstrate the advantages and limitations of the proposed method.

4.1 The Dirichlet distribution

The Dirichlet distribution is a generalization of the Beta distribution for multiple random variables. Let us consider a random variable $\mathbf{p} = \{p_1, \cdots, p_k\}$. The distribution is defined for all $\mathbf{p} \in \mathbb{S}$, where $\mathbb{S}$ is the standard simplex defined as $\{\mathbf{p} \in \mathbb{R}^k : p_i \geq 0, \sum p_i = 1\}$.

In other words, the elements of $\mathbf{p}$ must be positive and sum up to 1. For this reason, each p_i may be interpreted as a probability itself and the Dirichlet distribution is known as a distribution over probability distributions. Thus, it is commonly applied to estimate proportional data (Ng; Tian; Tang, 2011).

The Dirichlet probability density function (PDF) is defined by (Ng; Tian; Tang, 2011):

$$p(\mathbf{p}) \sim \mathfrak{D}(\mathbf{p}; \boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{i=1}^{k} \alpha_i\right)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} p_i^{\alpha_i - 1} \tag{4.1}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_k\}$ is the distribution's parameters and $\alpha_k > 0$. It is important to note that $\boldsymbol{\alpha}$ and $\mathbf{p}$ are both k-dimensional. In addition, $\boldsymbol{\alpha}$ does not need to be a distribution itself, but only a positive value. Γ is the Gamma function defined as $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$. In brief, this function is the generalization of the factorial function for any real value. It is described as $\Gamma(x) = (x-1)!$ with three main properties: $\Gamma(1) = 1$, $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(\frac{1}{2}) = \sqrt{\pi}$. Using these properties we can compute $\Gamma(x)$ for any $x \in \mathbb{R}$ (Sebah; Gourdon, 2002).

4.1.1 Expectation, variance and covariance

Considering a Dirichlet distribution $\mathfrak{D}(\mathbf{p}; \boldsymbol{\alpha})$, with $\mathbf{p} = \{p_1, \cdots, p_k\}$ and $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_k\}$, the expectation, variance, and covariance of this distribution are defined in Equations (4.2), (4.3), and (4.4), respectively (Ng; Tian; Tang, 2011).

$$E[p_i] = \frac{\alpha_i}{\sum_{j=1}^{k} \alpha_j} \tag{4.2}$$

$$Var[p_i] = \frac{\alpha_i \left(\sum_{i=1}^{k} \alpha_i - \alpha_i\right)}{\left(\sum_{i=1}^{k} \alpha_i\right)^2 \left(\sum_{i=1}^{k} \alpha_i + 1\right)} \tag{4.3}$$

$$Cov[p_i, p_j] = \frac{-\alpha_i \alpha_j}{\left(\sum_{i=1}^{k} \alpha_i\right)^2 \left(\sum_{i=1}^{k} \alpha_i + 1\right)}, \; i \neq j \tag{4.4}$$

with $i = 1, \cdots, k$.

4.1.2 Estimating a Dirichlet distribution

Let us consider a set of data $D = \{\mathbf{p}_1, \cdots, \mathbf{p}_N\}$, where each $\mathbf{p}_n = \{p_1, \cdots, p_k\}$, $\mathbf{p}_n \in \mathbb{S}$, $n = 1, \cdots, N$, and N is the number of samples in the set. The parameters $\boldsymbol{\alpha}$ of a Dirichlet distribution may be estimated from D using the Maximum Likelihood Estimation

(MLE) – see Section 2.1.4. The Dirichlet log-likelihood function of the data is given by (Minka, 2012):

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= \log p(D \mid \boldsymbol{\alpha}) = \log \prod_n p(\mathbf{p}_n \mid \boldsymbol{\alpha}) \\ &= \log \prod_n \frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma(\alpha_k)} \prod_k p_{nk}^{\alpha_k - 1} \\ &= N\left(\log \Gamma\left(\sum_k \alpha_k\right) - \sum_k \log \Gamma(\alpha_k) + \sum_k (\alpha_k - 1) \log \bar{p}_k\right)\end{aligned} \tag{4.5}$$

where $\log \bar{p}_k = \frac{1}{N} \sum_n \log p_{nk}$. Since the Dirichlet distribution belongs to the exponential family (Ronning, 1989), the objective function $\mathcal{L}(\boldsymbol{\alpha})$ is convex in $\boldsymbol{\alpha}$. Thus, it is guaranteed that the function has a unique optimum (Minka, 2012).

A simple way to maximize $\mathcal{L}(\boldsymbol{\alpha})$ is using the gradient ascent algorithm – see Section 2.2.2.1. To this end, the gradient of $\mathcal{L}(\boldsymbol{\alpha})$ is computed as:

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \frac{\partial \mathcal{L}(\boldsymbol{\alpha})}{\partial \alpha_k} = N\left(\Psi\left(\sum_k \alpha_k\right) - \Psi(\alpha_k) + \log \bar{p}_k\right) \tag{4.6}$$

where $\Psi = \frac{d \log \Gamma(x)}{dx}$ is known as Digamma function. From Equation (4.6), we can apply the gradient ascent algorithm, considering the restriction $\boldsymbol{\alpha} > 0$, to estimate the Dirichlet distribution for the set of data D. Nonetheless, Minka (2012) proposed a fixed-point iteration for maximizing the log-likelihood, and consequently estimate $\boldsymbol{\alpha}$, that is faster than gradient ascent method. The main task is to determine an initial value for $\boldsymbol{\alpha}$ and to find a lower bound function which is tight at $\boldsymbol{\alpha}$. Then, this function is optimized to find a new value of $\boldsymbol{\alpha}$. Thereby, let us consider the following inequality (Dragomir; Agarwal; Barnett, 2000):

$$\Gamma(x) \geq \Gamma(\hat{x}) e^{(x - \hat{x})\Psi(\hat{x})} \tag{4.7}$$

Applying this inequality to $\Gamma(\sum_k \alpha_k)$, the following lower bound on the log-likelihood is obtained (Minka, 2012):

$$\mathcal{L}(\boldsymbol{\alpha}) \geq N\left[\left(\sum_k \alpha_k\right) \Psi\left(\sum_k \hat{\alpha}_k\right) - \sum_k \log \Gamma(\alpha_k) + \sum_k (\alpha_k - 1) \log \bar{p}_k + \text{Const.}\right] \tag{4.8}$$

Next, one maximizes the previous equation by setting the gradient to zero and solving it for $\boldsymbol{\alpha}$, which leads to the following fixed-point iteration (Minka, 2012):

$$\alpha_k^{\text{new}} = \Psi^{-1}\left[\Psi\left(\sum_k \alpha_k^{old} + \log \bar{p}_k\right)\right] \tag{4.9}$$

As we can note in Equation 4.9, the optimization requires the inversion of Ψ. This is efficiently achieved by applying the Newton-Raphson update procedure (Ypma, 1995). Let us consider two arrays $\mathbf{a} = \{a_1, \cdots, a_k\}$ and $\mathbf{b} = \{b_1, \cdots, b_k\}$. It is possible to approximate the $\Psi^{-1}(\mathbf{a})$ applying the following update rule (Minka, 2012):

$$b_k^{\text{new}} = b_k^{\text{old}} - \frac{\Psi(b_k) - a_k}{\frac{\partial \Psi(\mathbf{b})}{\partial b_k}} \tag{4.10}$$

A starting point for $\mathbf{b}$ may be found using the following asymptotic equation:

$$b_k \approx \begin{cases} \log(a_k - \frac{1}{2}) & \text{if } a_k \geq -2.22 \\ -\frac{1}{a_k} - \gamma & \text{if } a_k < -2.22 \end{cases} \tag{4.11}$$

where $\gamma = \Psi(-1)$ is known as Euler–Mascheroni constant. Only five iterations of Equation 4.10 is enough to approximate $\Psi^{-1}(\mathbf{a})$ with 14 digits of precision. In Algorithm 2 is described the pseudo-code to the Dirichlet estimation via fixed-point iteration method. In Appendix B.4 we present some tests to assess the convergence time and error of this algorithm.

Algorithm 2: *Dirichlet estimation using fixed-point iteration method*

Input:
    $D = \{\mathbf{p}_1, \cdots, \mathbf{p}_N\}$ with $\mathbf{p} = \{p_1, \cdots, p_k\}$ and $\mathbf{p} \in \mathbb{S}$;
    A tolerance *tol* and maximum iterations *max_iter*;
$D_{log} = \frac{1}{N} \sum_{n,k} \log p_{nk}$;
$\alpha_{old} = \frac{1}{N} \sum_{n,k} p_{nk}$;
while $j < max\_iter$ **do**:
    Compute α^{new} according to (4.9) and using b^{new}(4.10);
    Compute $\mathcal{L}(\alpha^{\text{old}})$ and $\mathcal{L}(\alpha^{\text{new}})$ according to (4.5);
    if $\mathcal{L}(\alpha^{\text{old}}) - \mathcal{L}(\alpha^{\text{new}}) < tol$:
        break;
    $\alpha^{\text{old}} = \alpha^{\text{new}}$;
    $j = j + 1$;
return α_{new};

4.2 Methodology

In this section, we describe our method to **le**arn dynamic **w**eights for an ensemble of deep learning models based on the **Dir**ichlet distribution (LeWDir). The main idea behind this method is to learn the probability distribution of the deep models' predictions for a set of unseen samples and compute a score that represents the relevance of each of them within an ensemble. We describe the method in two steps. First, we describe the probability distribution estimation of the validation set predictions. Second, we detail the weights computation.

4.2.1 Step 1: Estimating the probability distribution

Let us consider a classification problem with $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ inputs, $Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$ outputs, and $\mathbf{c} = \{c_1, \cdots, c_L\}$ labels, where N and L are the number of samples and labels, respectively. The first step to train a deep model on this task is to split X and Y into training (X_{tr} and Y_{tr}) and validation (X_{val} and Y_{val}) partitions. The model is trained using $\{X_{\text{tr}}, Y_{\text{tr}}\}$ to fit a function $f(\mathbf{x})$ that outputs an array $\mathbf{y}$ containing the logits for the sample $\mathbf{x}$ to each label c. Usually, the logit is converted to the probability of y to assume a label c, i.e, $p(y = c \mid \mathbf{x})$. The most common way to assign this probability is through the *softmax* function:

$$p_l(y_l = c_l \mid \mathbf{x}) = \frac{e^{y_l}}{\sum_{z=1}^{L} e^{y_z}}, \quad l = 1, \cdots, L \tag{4.12}$$

Therefore, when a new sample $\check{\mathbf{x}}$ is presented to f, it outputs an array of probabilities $\mathbf{p}$ that represents the confidence of this sample for each label $c_l \in \mathbf{c}$. In such case, we can interpret each array $\mathbf{p}$ as a multivariate random variable and apply the fixed-point method, described in Algorithm 2, to estimate the Dirichlet distribution of a given partition of X. In particular, our goal is to estimate the Dirichlet distribution for the following sets of probabilities:

- P_l^h: the set of probabilities obtained by the model to the validation set considering the label $c_l \in \mathbf{c}$ when the label is correctly predicted by the network. We name it the hit set for the label l.
- P_l^m: the set of probabilities obtained by the model to the validation set considering the label $c_l \in \mathbf{c}$ when the label is incorrectly predicted by the network. We name it the miss set for the label l.

In order to sample in the model space, we applied the dropout sampling (Gal; Ghahramani, 2016) for the validation partition. The dropout sampling consists of performing the model with the dropout rate (Srivastava et al., 2014) active during the inference phase. Normally, the dropout is active only during the training phase as a tool to avoid overfitting and increase generalization (Gal; Ghahramani, 2016). The goal of activating it also during inference is to vary the model and assess how well the model generalizes for unseen samples. In our case, the sets P_l^h and P_l^m are collected after d_e executions of the network using dropout sampling with dropout rate equal to d_r. The parameter d_e controls the number of times that a given sample $\check{\mathbf{x}}$ is presented to the model; while the parameter d_r is the rate of nodes in the model that will be active during the execution. Therefore, if the model is well trained the output probabilities should not present a high variation for the same sample.

4.2.2 Step 2: Computing the dynamic weights

Let us consider an ensemble with G deep models $E = \{f_1(\mathbf{x}), \cdots, f_G(\mathbf{x})\}$ trained using $\{X_{\text{tr}}, Y_{\text{tr}}\}$. For each model $f_g(\mathbf{x})$, we apply the step 1, i.e, we perform the dropout sampling for X_{val} and estimate the Dirichlet distributions $\mathfrak{D}_g(P_l^h; \boldsymbol{\alpha}_l^h)$ and $\mathfrak{D}_g(P_l^m; \boldsymbol{\alpha}_l^m)$, i.e., the distributions for each P_l^h and P_l^m sets. We name them as the hit and miss distributions for the model g and label l, respectively.

Now, let us consider $\check{\mathbf{p}}_g$ the predictions obtained to a new sample $\check{\mathbf{x}}$ by a model f_g. The main goal of this step is to measure how far $\check{\mathbf{p}}_g$ is from the hit and miss distributions for each label l. In particular, we observed that a reliable prediction will be closer to a hit and far from a miss. An effective method to measure the distance between a random variable $\mathbf{z}$ from its distribution $P(\mathbf{z})$ is the Mahalanobis distance, defined as (Maesschalck; Jouan-Rimbaud; Massart, 2000):

$$Mah(\mathbf{z}, P(\mathbf{z})) = \sqrt{(\mathbf{z} - \mathbb{E}[P(\mathbf{z})])^T S^{-1} (\mathbf{z} - \mathbb{E}[P(\mathbf{z})])} \tag{4.13}$$

where $\mathbb{E}[P(\mathbf{z})]$ is the expectation value of the distribution $P(\mathbf{z})$ and S^{-1} is the inverse of the covariance matrix of the distribution. It is effective because it takes into account not only the expectation but also the variance of the distribution. Therefore, we apply the Mahalanobis distance to measure the difference between $\check{\mathbf{p}}_g$ and the distributions $\mathfrak{D}_g(P_l^h; \boldsymbol{\alpha}_l^h)$ and $\mathfrak{D}_g(P_l^m; \boldsymbol{\alpha}_l^m)$. The Dirichlet expectation and covariance matrix are computed using (4.3) and (4.4), and (4.2), respectively.

The standard way to select the label that the model f_g assigns to a sample $\check{\mathbf{x}}$ is through argmax($\check{\mathbf{p}}_g$). However, we interpret $\check{\mathbf{p}}_g$ as a multivariate random variable. Thus, to compute the Mahalanobis distance between $\check{\mathbf{p}}_g$ and the hit and miss distributions, we take into account the contribution of each label l according to the probability assigned to it. This concept is described by (4.14) and (4.15).

$$Mah_{f_g}^h = \sum_{l=1}^{L} \check{p}_{gl} Mah(\check{\mathbf{p}}_g, \mathfrak{D}_g(P_l^h; \boldsymbol{\alpha}_l^h)),\ g = 1, \cdots, G \tag{4.14}$$

$$Mah_{f_g}^m = \sum_{l=1}^{L} \check{p}_{gl} Mah(\check{\mathbf{p}}_g, \mathfrak{D}_g(P_l^m; \boldsymbol{\alpha}_l^m)),\ g = 1, \cdots, G \tag{4.15}$$

Finally, we compute the relevance of the model f_g for a new sample $\check{\mathbf{x}}$ as follows:

$$w_{f_g} = \frac{Mah_{f_g}^m}{Mah_{f_g}^h},\ g = 1, \cdots, G \tag{4.16}$$

As we may note, this simple equation implements the idea previously mentioned. If $\check{\mathbf{p}}_g$ is close to the hit distribution and far from the miss one, the weight increases. On the other

hand, if $\check{\mathbf{p}}_g$ is near to the miss and far from the hit, the weight decreases. To normalize the weights in the interval $[0, 1]$, we apply the following equation:

$$\tilde{w}_{f_g} = \frac{w_{f_g}}{\sum_{j=1}^{G} w_{f_j}},\ g = 1, \cdots, G \tag{4.17}$$

The last step of the proposed method is to compute the final aggregation according to the following equation:

$$agg_{f_g} = \sum_{j=1}^{L} \tilde{w}_{f_g} \check{p}_j,\ g = 1, \cdots, G \tag{4.18}$$

It is worth noticing that the learned weights may be used by any aggregation function.

In Figure 28 is illustrated a schematic diagram and in Algorithm 3 is described the pseudocode of the proposed method. It is important to note that only the step 2 is computed online. As shown in the figure, the Dirichlet estimations for each model are saved during step 1. As such, the relevance of each model in the ensemble is computed online for a new unseen sample $\check{\mathbf{x}}$, which means that the weights are dynamic and change according to the presented sample. In this context, using this method we can identify weak models according to the sample. In addition, it is possible to prune the ensemble online by selecting the best g models for each new sample or setting a threshold for the weights. In the next section, we present an illustrative example of the proposed method.

Algorithm 3: *Ensemble aggregation based on LewDir*

Input:
    An ensemble of G models $E = \{f_1(\mathbf{x}), \cdots, f_G(\mathbf{x})\}$;
    A validation partition $X_{\text{val}} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$;
For each model $f_i \in E$ **do**:
    Apply Algorithm 2 using X_{val} to estimate $\mathfrak{D}_{f_i}(P_l^h; \boldsymbol{\alpha}_l^h)$ and $\mathfrak{D}_{f_i}(P_l^m; \boldsymbol{\alpha}_l^m)$;
For a new sample $\check{\mathbf{x}}$ **do**:
    For each model $f_i \in E$ **do**:
        Compute $Mah_{f_i}^h$ and $Mah_{f_i}^m$;
        Compute the normalized weights $\tilde{w}_i$;
        Compute the final aggregation g_{f_i};
return: aggregation array $\mathbf{g} = \{g_{f_i}, \cdots, g_{f_M}\}$

4.2.3 Illustrative example

In this section, we present a numerical example to show how the proposed method to assign dynamic weights for an ensemble works. First, let us consider the six deep models used in Section 3.4 to classify the ISIC 2019 dataset. These models, i.e., EfficientNet-B1 and B4, DenseNet-121, MobileNet-V2, ResNet-50, and VGGNet-13, are used to compose an ensemble to classify the same dataset.

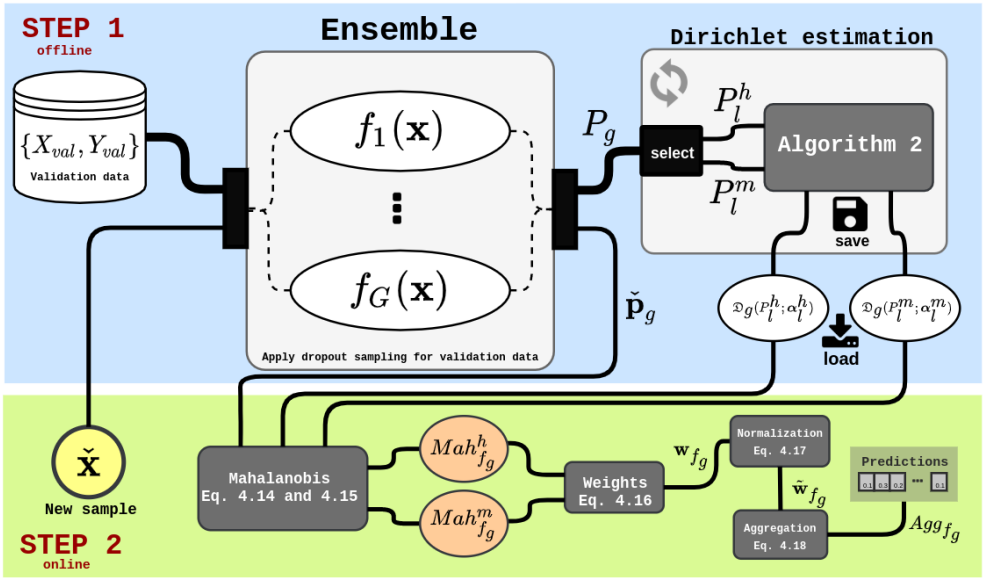


Figure 28 – A schematic diagram illustrating both steps of the proposed algorithm LewDir. In the first step, the validation data is used to get and save the hit and miss distributions. In the second step, the distributions are loaded and used to estimate the weights for each model within the ensemble according to the probabilities assigned to the new sample.

Let us consider that a new unseen sample – in which the ground truth label is NV – is evaluated by the ensemble and each model assigned the probabilities presented in Table 7:

Model	AK	BCC	BKL	DF	MEL	NV	SCC	VASC
EfficientNet-B1	0.000	0.001	0.003	0.003	0.093	0.898	0.000	0.002
EfficientNet-B4	0.000	0.001	0.012	0.000	0.291	0.691	0.001	0.002
DenseNet-121	0.000	0.000	0.005	0.000	0.544	0.450	0.000	0.000
MobileNet-v2	0.001	0.002	0.005	0.003	0.188	0.797	0.001	0.002
Resnet-50	0.000	0.001	0.004	0.000	0.067	0.927	0.000	0.001
VGG-13	0.000	0.000	0.004	0.000	0.155	0.840	0.000	0.000

Table 7 – Probabilities assigned by each model to a new sample. The true label is NV.

Now, we use the Mahalanobis distance, described in Equation 4.13, to compute the distance between each models' predictions to the miss and hit distributions according to Equations 4.14 and 4.15. For the models and prediction in Table 7, the distances returned for the miss and hit are:

$$\begin{aligned} \mathbf{d}_{\text{miss}} &= [0.106, 0.250, 0.291, 0.187, 0.076, 0.145] \\ \mathbf{d}_{\text{hit}} &= [0.339, 0.215, 0.189, 0.275, 0.346, 0.400] \end{aligned} \tag{4.19}$$

Next, we compute the relevance of each model within the ensemble according to Equation

Model	AK	BCC	BKL	DF	MEL	NV	SCC	VASC
AVG	0.000	0.001	0.005	0.001	0.223	0.767	0.000	0.001
MAX	0.001	0.002	0.008	0.002	0.365	0.620	0.001	0.002
PROD	0.000	0.000	0.000	0.000	0.010	0.990	0.000	0.000
MV	0.000	0.000	0.000	0.000	0.167	0.833	0.000	0.000
SWA	0.000	0.001	0.005	0.001	0.225	0.766	0.000	0.001
NP-AVG	0.000	0.001	0.006	0.001	0.216	0.774	0.000	0.001
LewDir	0.000	0.001	0.004	0.001	0.142	0.85	0.000	0.001

Table 8 – The probabilities aggregated, after normalization, by each method to each label in ISIC 2019 dataset.

4.16, which results in the following scores:

$$\mathbf{w} = [3.194, 0.859, 0.649, 1.473, 4.547, 2.753] \tag{4.20}$$

Applying the normalization described in Equation 4.17, we obtain the final weights:

$$\tilde{\mathbf{w}} = [0.237, 0.064, 0.048, 0.109, 0.337, 0.204] \tag{4.21}$$

As we can see, the method assigned the highest weights to EfficientNet-B1, ResNet-50, and VGG-13, and the models that returned the highest probabilities for the correct label. Now, we may use these weights to compute the aggregated probability according to Equation 4.18.

Let us consider the four most common aggregation operators: majority voting (MV), predictions average (AVG), maximum prediction (MAX), and product of predictions (PROD). Let us also consider the static weights average (SWA) proposed by Harangi (2018) and the untrained dynamic weighting approach NP-AVG/NP-MAX proposed by Krawczyk and Woźniak (2016) with standard deviation equal to 1. In Table 8 is presented the aggregated probabilities assigned by LewDir and each of these aggregation methods for this specific example. After the aggregation, we normalize all values to the interval $[0, 1]$. As we can see, all aggregation methods resulted in the correct label. In addition, the highest probabilities assigned to the correct label are produced by PROD and LewDir. The PROD method performs well for this example because there is no strong disagreement in the ensemble. However, it is fairly frequent to get a disagreement, and when this happens, the method does not present the same performance. On the other hand, LewDir is more stable in terms of assigned probabilities. Usually, it improves the aggregated probabilities when compared to the other methods, as we can observe from the Table 8.

4.3 Experimental results

In this section, we carry out a set of experiments to evaluate the performance of the proposed method. First, we describe the setup of the experiments, including the datasets and deep learning models applied. Next, we show the results and compare them with standard approaches used in medical imaging tasks. Lastly, we present a discussion about the results.

4.3.1 Experiments setup

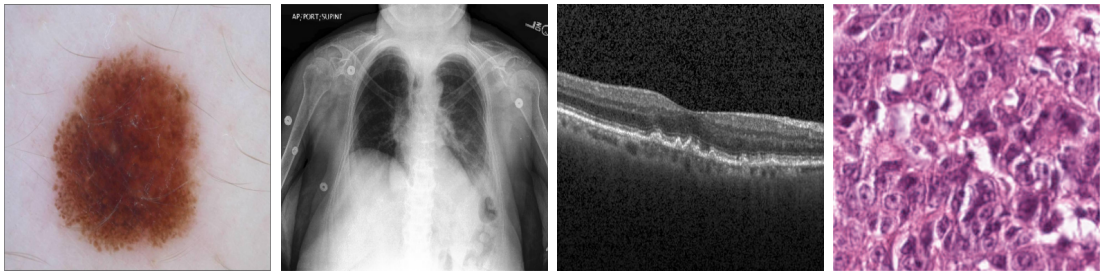
In order to test our method, we create an heterogeneous ensemble composed of six well-known CNN architecures: DenseNet-121 (Huang et al., 2017b), GoogleNet (Szegedy et al., 2015), InceptionV4 (Szegedy et al., 2017), MobileNet-v2 (Sandler et al., 2018), ResNet-50 (He et al., 2016), and VGG-16 (Simonyan; Zisserman, 2014). The ensemble was trained on four state-of-the-art medical datasets:

- **CheXpert** (Irvin et al., 2019): a large chest radiograph dataset containing 224,316 images reporting the presence of different conditions in radiology. For this work, we select the pleural effusion condition considering the U-MultiClass, which has 3 classes: positive, negative, or uncertainty of the condition.
- **NCT**-CRC-HE-100K (Kather et al., 2019): a set of 100,000 histological images of human colorectal cancer (CRC), and normal tissue.
- **OCT** (Kermany et al., 2018): a dataset containing 83,495 images of retinal optical coherence tomography (OCT) with 4 labels, 3 diseases, and a normal retina.

In addition to this list of datasets, we also include the **ISIC** 2019 dataset; however, as this dataset has images and meta-data, we create an ensemble using the models that we evaluate for the MetaBlock approach in Section 3.4. In Figure 29 is depicted an image sample of each dataset previously described.

All models in the ensemble were trained using their original architecture, except for the last layer that depends on the number of labels in the dataset. The training phase is carried out using the same configuration we describe in Section 3.4.1, i.e., using SGD optimizer along with weight decay, momentum, and early stopping. Likewise the ISIC dataset, the rest of the datasets are also imbalanced. Thus, we also applied the weighted cross-entropy as the loss function in which the weights are determined according to the labels' frequency. All images were resized to 224×224 and we applied a data augmentation using simple image processing operations such as flips and scaling.

For all experiments, we also reserved approximately 10% of each dataset for testing respecting the labels' frequency distribution. The remaining 90% of data were used on



(a) ISIC (b) CheXpert (c) OCT (d) NCT

Figure 29 – An example of an image from each medical dataset used in this experiment. We may observe that they have different features that affect the level of difficulty of each task.

the training phase through 5-fold cross-validation for assessing the effectiveness of the models. For each folder, we trained the models and estimate $\mathfrak{D}_g(P_l^m; \boldsymbol{\alpha}_l^m)$ and $\mathfrak{D}_g(P_l^m; \boldsymbol{\alpha}_l^m)$ as described in section 4.2. To measure the performance, we computed the average and standard deviation of the following metrics: accuracy (ACC), balanced accuracy (BACC), aggregated area under the curve (AUC), and the cross-entropy (Loss). We compared the performance of our method with the other six aggregation methods described in the illustrative example section. We also perform the Friedman test following by the Wilcoxon test (if applicable), using $p_{\text{value}} = 0.05$, and the A-TOPSIS ranking algorithm (Krohling; Pacheco, 2015) to assess the results. All procedures involving CheXpert, OCT, and NCT were implemented using Python and PyTorch and performed on Nvidia Tesla P-100.

4.3.2 Experiment results

In this section, we present the performance obtained by each deep model and the ensemble for the medical imaging datasets previously described. We present the individual results of each model for CheXpert, NCT, and OCT. The results for ISIC – considering the MetaBlock approach – is already described in Table 3.

In Table 9 is reported the performance for each deep model and each dataset. As we may note, there are different levels of performance among the datasets. While the deep models present BACC over 90% for NCT and OCT, they do not present the same performance for ISIC and, especially, for CheXpert. In fact, as we can see through the loss metric, the models struggle to generalize to these latter datasets. It happens mainly because they present a high variability of features and strong similarities among diseases. For instance, in ISIC it is hard to distinguish melanoma and nevus (Gessert et al., 2019), while in CheXpert it is common to confuse the uncertainty label between positive and negative (Irvin et al., 2019).

Overall, due to the imbalance among the labels, the BACC is slightly lower than ACC for ISIC, NCT, and OCT. For CheXpert, it is approximately 10% lower. The reason

Model	CheXpert			
	ACC	**BACC**	**AUC**	**Loss**
DenseNet-121	0.676 ± 0.002	0.560 ± 0.006	0.781 ± 0.002	0.824 ± 0.015
GoogleNet	0.668 ± 0.002	0.561 ± 0.002	0.788 ± 0.001	0.830 ± 0.001
InceptionV4	0.656 ± 0.015	0.555 ± 0.003	0.768 ± 0.005	0.837 ± 0.015
MobileNet-V2	0.684 ± 0.008	0.549 ± 0.002	0.774 ± 0.001	0.817 ± 0.008
Resnet-50	0.685 ± 0.006	0.568 ± 0.009	0.794 ± 0.006	0.803 ± 0.010
VGG-16	0.675 ± 0.014	0.561 ± 0.020	0.789 ± 0.016	0.825 ± 0.010
Model	**NCT**			
	ACC	**BACC**	**AUC**	**Loss**
DenseNet-121	0.934 ± 0.014	0.909 ± 0.032	0.992 ± 0.001	0.403 ± 0.078
GoogleNet	0.924 ± 0.008	0.908 ± 0.003	0.989 ± 0.006	0.464 ± 0.191
InceptionV4	0.903 ± 0.005	0.883 ± 0.013	0.984 ± 0.006	0.502 ± 0.006
MobileNet-V2	0.935 ± 0.007	0.924 ± 0.001	0.993 ± 0.002	0.354 ± 0.023
Resnet-50	0.926 ± 0.004	0.920 ± 0.001	0.992 ± 0.001	0.442 ± 0.023
VGG-16	0.914 ± 0.014	0.911 ± 0.008	0.995 ± 0.002	0.392 ± 0.077
Model	**OCT**			
	ACC	**BACC**	**AUC**	**Loss**
DenseNet-121	0.976 ± 0.002	0.956 ± 0.002	0.999 ± 0.000	0.064 ± 0.004
GoogleNet	0.968 ± 0.007	0.944 ± 0.013	0.997 ± 0.002	0.095 ± 0.026
InceptionV4	0.976 ± 0.001	0.957 ± 0.002	0.998 ± 0.000	0.073 ± 0.002
MobileNet-V2	0.957 ± 0.003	0.943 ± 0.006	0.995 ± 0.001	0.132 ± 0.017
Resnet-50	0.972 ± 0.002	0.952 ± 0.006	0.998 ± 0.000	0.081 ± 0.008
VGG-16	0.978 ± 0.002	0.960 ± 0.003	0.999 ± 0.001	0.065 ± 0.007

Table 9 – Performance achieved by each CNN architecture individually for each medical image dataset used in this experiment.

for this difference in performance is that the models are not identifying the uncertainty label properly. Thereby, we also consider the BACC as the priority metric in this experiment. Actually, this is the main metric for several medical task challenges such as ISIC (ISIC, 2019).

It is worth mentioning that there is no model that presents the best performance for all datasets. Considering the average BACC, for ISIC and CheXpert the best performance is presented by ResNet-50, for NCT by MobileNet-V2, and for OCT by VGG-16. This result is in line with the *No Free Lunch theory* (Wolpert, 1996) and supports the use of an ensemble of models as an approach to improve the performance and robustness of deep learning models' performance.

In Table 10 is presented the results achieved by the ensemble of CNN models, considering the six aggregation methods and our proposed approach, for each medical imaging dataset. In general, the aggregation methods present better performance than the single models. We highlight the following observations for each dataset:

- ISIC: for this dataset, our method improved the average of the BACC in 4.2% comparing to the best single model and almost 1% comparing to the second-best

Method	ISIC 2019			
	ACC	**BACC**	**AUC**	**Loss**
AVG	0.824 ± 0.002	0.800 ± 0.005	0.970 ± 0.002	0.551 ± 0.011
MAX	0.814 ± 0.010	0.807 ± 0.040	0.971 ± 0.003	0.610 ± 0.028
PROD	0.826 ± 0.011	0.798 ± 0.080	$\mathbf{0.975 \pm 0.002}$	1.464 ± 0.048
MV	0.808 ± 0.040	0.790 ± 0.005	0.945 ± 0.003	2.580 ± 0.130
SWA	0.821 ± 0.003	0.805 ± 0.011	0.972 ± 0.002	0.552 ± 0.012
NP-AVG	0.823 ± 0.002	0.799 ± 0.005	0.974 ± 0.002	0.527 ± 0.010
LewDir	$\mathbf{0.831 \pm 0.004}$	$\mathbf{0.813 \pm 0.004}$	0.973 ± 0.002	$\mathbf{0.512 \pm 0.015}$
Method	**NCT**			
	ACC	**BACC**	**AUC**	**Loss**
AVG	0.941 ± 0.002	0.928 ± 0.002	0.996 ± 0.001	0.189 ± 0.015
MAX	0.940 ± 0.003	0.931 ± 0.003	0.995 ± 0.002	0.201 ± 0.013
PROD	0.942 ± 0.002	$\mathbf{0.932 \pm 0.003}$	0.995 ± 0.000	1.167 ± 0.051
MV	$\mathbf{0.943 \pm 0.001}$	0.931 ± 0.003	0.984 ± 0.002	0.782 ± 0.092
SWA	0.932 ± 0.002	0.922 ± 0.002	0.996 ± 0.001	0.196 ± 0.003
NP-AVG	0.942 ± 0.002	0.931 ± 0.002	0.996 ± 0.001	$\mathbf{0.187 \pm 0.003}$
LewDir	0.942 ± 0.001	$\mathbf{0.932 \pm 0.004}$	0.996 ± 0.002	0.251 ± 0.002
Method	**CheXpert**			
	ACC	**BACC**	**AUC**	**Loss**
AVG	0.697 ± 0.003	0.571 ± 0.004	0.801 ± 0.003	0.800 ± 0.003
MAX	0.715 ± 0.007	0.566 ± 0.002	0.788 ± 0.003	0.819 ± 0.001
PROD	0.695 ± 0.003	$\mathbf{0.574 \pm 0.005}$	0.797 ± 0.004	1.469 ± 0.014
MV	0.709 ± 0.001	0.567 ± 0.011	0.760 ± 0.006	3.689 ± 0.124
SWA	0.695 ± 0.003	0.572 ± 0.004	0.790 ± 0.006	0.801 ± 0.004
NP-AVG	0.702 ± 0.003	0.572 ± 0.005	0.793 ± 0.005	0.801 ± 0.004
LewDir	$\mathbf{0.724 \pm 0.002}$	0.568 ± 0.003	$\mathbf{0.801 \pm 0.005}$	$\mathbf{0.728 \pm 0.004}$
Method	**OCT**			
	ACC	**BACC**	**AUC**	**Loss**
AVG	0.980 ± 0.002	0.962 ± 0.002	0.999 ± 0.002	0.064 ± 0.004
MAX	0.979 ± 0.002	0.960 ± 0.003	0.999 ± 0.001	0.095 ± 0.003
PROD	0.980 ± 0.002	0.960 ± 0.004	0.998 ± 0.001	0.190 ± 0.033
MV	0.980 ± 0.003	0.962 ± 0.005	0.995 ± 0.001	0.198 ± 0.011
SWA	0.980 ± 0.002	0.961 ± 0.005	0.998 ± 0.000	0.063 ± 0.003
NP-AVG	0.99 ± 0.002	0.966 ± 0.002	0.999 ± 0.002	0.036 ± 0.004
LewDir	$\mathbf{0.994 \pm 0.00}2$	$\mathbf{0.972 \pm 0.002}$	0.999 ± 0.000	$\mathbf{0.029 \pm 0.001}$

Table 10 – Performance achieved by the ensemble of CNNs for each medical image dataset considering the different aggregation methods. In bold is highlighted the highest average for each metric.

aggregation method. In addition, it presents the lowest loss, which means it is predicting the correct label with more confidence.

- NCT: the results obtained by the ensemble are slightly better than the single models and all aggregation methods present similar performance for ACC and BACC. Nonetheless, for this dataset, the lowest average loss is achieved by the NP-AVG approach.

- CheXpert: our method improved the average ACC in 3.9% comparing to the best single model. However, the BACC achieved by the ensemble is quite similar to the single models. Again, our approach presents the lowest loss among all aggregation methods.

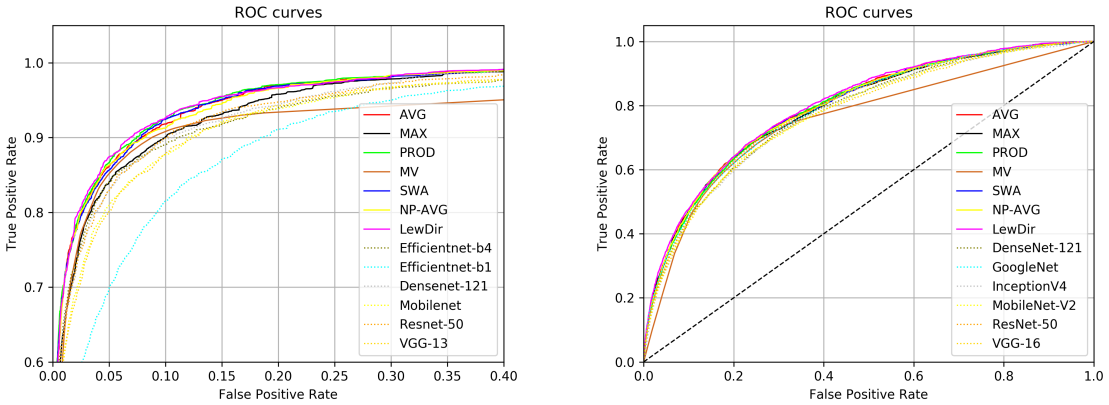
- OCT: our method improved the average BACC in 1.2% comparing to the best single model and almost 1% comparing to the second-best aggregation method. As the models perform quite well for this dataset, the remaining metrics are quite similar.

Overall, our method presents the lowest average loss for 3 out of 4 datasets and best average BACC for ISIC and OCT. Nonetheless, AVG, SWA, and NP-AVG present competitive performance considering all four datasets. We also observe that the ensemble improves the AUC only for ISIC and CheXpert. Lastly, it is worth to note that the PROD and MV present the highest average loss among all methods. It happens because whenever there is a disagreement in the ensemble, these methods allow it to impact too much in the aggregation prediction.

In Figure 30 is shown the receiver operating characteristic (ROC) curves considering one folder for ISIC and CheXpert datasets. As we may see, the aggregation methods present competitive performance, although our method's curve is slightly above the others. In general, the deep models' curve is below the aggregations' ones. In addition, from the ROC curve for the CheXpert dataset, we observe that the ensemble does not present the same performance as the one presented to the ISIC 2019 dataset.

To compare the aggregation methods, first, we performed the Friedman test, which returned $p_{\text{value}} = 0.0001$. Thus, we performed the Wilcoxon test, which returned 21 pairwise comparisons[2]. In summary, the test pointed out that LewDir, SWA, AVG, and NP-AVG are statistically different from the others. The p_{value} returned to LewDir when compared to AVG, SWA, and NP-AVG is slightly below the defined threshold – approximately 0.045 for all pairs. To help visualization, in Figure 31 is depicted the aggregation approaches rank generated by A-TOPSIS. As we can note, the ranking is close to the statistical test and the AVG, NP-AVG, and LewDir are too close in the rank.

[2] As there are too many comparisons, we summarize the test results without showing all p_{value}



(a) ISIC (b) CheXpert

Figure 30 – The macro average ROC curves for the ISIC and CheXpert datasets considering all deep models and aggregation approaches.

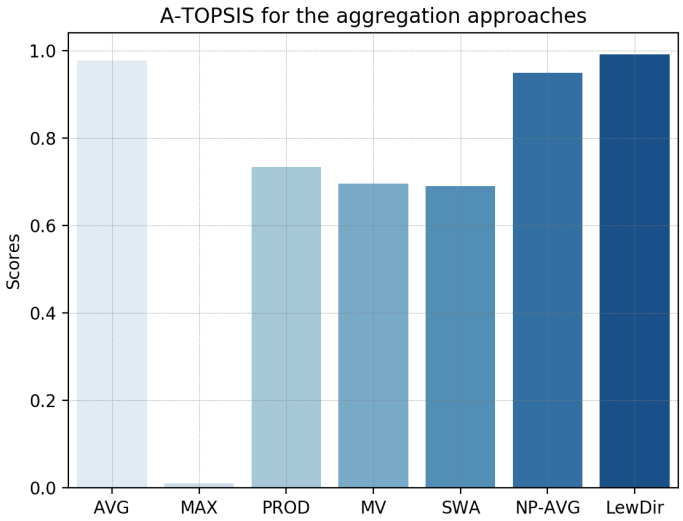


Figure 31 – The A-TOPSIS rank for the seven aggregation methods considering the BACC metric.

To conclude this experiment, although LewDir is not designed focusing on ensemble selection, we test it to online select CNN models from the pool based on the weights assigned to each model. For each new sample evaluated by the ensemble, we selected the best models through ranking the weights. We performed this experiment using only the ISIC dataset since the deep models present different results for this set. In Table 11 is presented the results of the ensemble selection considering from 1 to 5 models from the whole group. As the weights change according to the sample, the models selected may change every time the ensemble evaluated a new sample.

As we may note from Table 11, the results present a similar performance for all configurations, which suggests the weights assigned by the proposed approach are working

Nº of models	**ACC**	**BACC**	**AUC**	**Loss**
1	0.807 ± 0.010	0.813 ± 0.009	0.962 ± 0.002	0.679 ± 0.030
2	0.811 ± 0.010	0.814 ± 0.006	0.968 ± 0.001	0.592 ± 0.019
3	0.813 ± 0.008	0.813 ± 0.004	0.970 ± 0.001	0.558 ± 0.018
4	0.816 ± 0.006	0.815 ± 0.005	0.971 ± 0.001	0.540 ± 0.016
5	0.817 ± 0.006	0.814 ± 0.003	0.972 ± 0.001	0.523 ± 0.016
All	0.831 ± 0.004	0.813 ± 0.004	0.973 ± 0.002	0.512 ± 0.015

Table 11 – Ensemble selection according to the weights assigned by our method for the ISIC 2019 dataset

properly. In addition, we observe that the loss decreases as the number of selected models increases. This result shows that even though it is possible to achieve similar performance, in terms of BACC, by pruning the ensemble, the number of selected models is directly proportional to the prediction confidence since the Loss decreases when the number of selected models increases.

4.3.3 Discussion

The results presented in this section show that an ensemble of deep models worked properly to deal with different medical imaging classification. Although the computational burden to train several deep models is high, the improvement presented is desired, in particular for medical tasks. However, it is important to point out that in some cases using an ensemble is not feasible due to memory and/or processing limitations. For example, a CAD system embedded in a smartphone or in limited hardware such as a Raspberry Pi, may not be able to run an ensemble locally. Essentially, employing or not an ensemble of deep models is a trade-off between computational resources and performance.

In general, the proposed approach LewDir presented a competitive performance compared to the other methods, in particular to the NP-AVG, which also determines weights for the ensemble. It is worth mentioning that the NP-AVG assigns weights for each model and each label while LewDir considers only the models. Beyond to achieve the best average BACC for 3 out of 4 datasets, the method increases the confidence of the predictions since it presented the lowest loss for 3 out of 4 datasets. The advantage of providing online weights for each new evaluated sample is important towards understanding the real contribution of each model within the ensemble. In addition, it can be applied to select models in the pool, as shown in the experiments.

Despite promising results, we point out a limitation in LewDir. As it depends on the validation set to estimate the Dirichlet distributions of the hit and miss sets, the model is sensitive to this partition. For instance, if the distribution of this set is too far from the test set, i.e., the real world, the approach performance may decrease. However, deep learning training also depends on this assumption, i.e., the data distribution used in the

training phase is close to the one in which the model will be applied – see Section 2.1.3. In any case, it is important to ensure a validation partition that represents well the problem.

5 Skin cancer detection based on clinical images and patient demographics - A case study

In this chapter, we present a case study about skin cancer detection using clinical images collected from smartphone and patient demographics. First, we describe the dataset we collected along with the Dermatological and Surgical Assistance Program (PAD) at the Federal University of Espírito Santo (UFES). Next, we perform experiments combining both images and patient demographics – using the methods described in Chapter 3. Finally, we create an ensemble of deep models and aggregate them using the method proposed in Chapter 4.

5.1 PAD-UFES-20 dataset

The Dermatological and Surgical Assistance Program (PAD) at the Federal University of Espírito Santo (UFES) is a nonprofit program that provides free skin lesion treatment, in particular, to low-income people who cannot afford private treatment. This is a full skin lesion treatment, from the screening to the surgical process (if needed), in 11 different countryside cities in Espírito Santo state. For historical reasons, the Espírito Santo state has received thousands of immigrants from Europe throughout the 19th century. As Brazil is a tropical country, most of these immigrants and their descendants were/are not adapted to this climate. As a result, there is a high incidence of skin lesions/cancer in this state and the PAD plays a fundamental role to assist these people (Frasson et al., 2017).

In late 2017, the Nature Inspired Computing Laboratory (LABCIN-UFES) and the PAD started a partnership that resulted in the creation of a web-based platform and a multi-platform smartphone application to collect and store patient clinical data and skin lesion images – a description of these software is available in Appendix A. In this section, we first detail the data collection workflow. Next, we describe the PAD-UFES-20 dataset (Pacheco et al., 2020a). Finally, we present an exploratory data analysis to get a gist from the meta-data.

5.1.1 Data collection

The data collection workflow is summarized in Figure 32. First of all, the patients have an appointment with a group of up to three senior dermatologists (at least 15 years

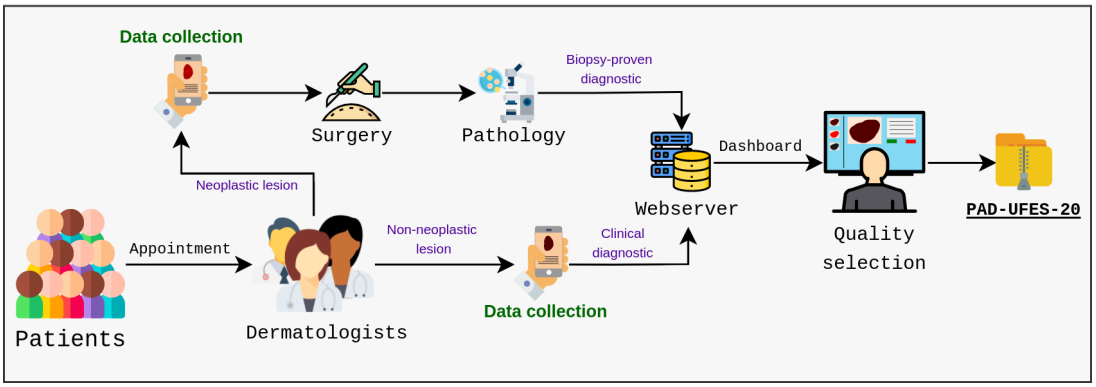


Figure 32 – Data collection workflow of the PAD-UFES-20 dataset from the clinical field to the quality selection.

of experience) that assesses the skin lesion. If the group identifies a neoplasm, the skin lesion is removed through surgical procedure – performed by medical students under the supervision of two senior plastic surgeons of the PAD – and sent to the Pathological Anatomy Unit of the University Hospital Cassiano Antônio Moraes (HUCAM) at the UFES to perform histopathology examination. On the other hand, if the group has a consensus that there is no neoplasm, they do not request a biopsy. In both cases, we collect images and clinical data. Later, when the biopsy result is available, it is filled for those lesions in which it was requested. All data is stored in a web-server and the final step is a quality selection to review every single sample that was collected in the previous steps.

The dataset was collected during 2018 and 2019. In total, there are over 50 types of skin lesions that were collected during this period. However, most of them are rare and contain only a few samples. For this reason, we selected the seven most common skin lesions diagnosed at PAD, which are: Basal Cell Carcinoma (BCC), Squamous Cell Carcinoma (SCC), Actinic Keratosis (ACK), Seborrheic Keratosis (SEK), Bowen's disease (BOD), Melanoma (MEL), and Nevus (NEV). As the Bowen's disease is considered SCC in situ (Wolff et al., 2017), we clustered them together, which results in six skin lesions[1] in the dataset, three skin cancers (BCC, SCC, and MEL) and three skin diseases (ACK, NEV, and SEK). The number of samples of each skin lesion is presented in Table 12. As we can see, the dataset is imbalanced, in particular for melanoma, the deadliest case of skin cancer. Unfortunately, imbalanced datasets are quite common for skin lesion datasets (Tschandl; Rosendahl; Kittler, 2018; Combalia et al., 2019), and finding solutions to deal with this issue is part of the skin cancer detection task.

Concerning the skin lesion diagnostic, for NEV, SEK, and ACK, only the clinical diagnosis is performed according to the PAD's dermatologists consensus during the patient's appointment. As these skin lesions are benign, it is not necessary to perform a skin biopsy. On the other hand, for skin lesions that dermatologists suspect malignancy, a biopsy is

[1] We name wounds, moles or spots on the skin as skin lesions. After the diagnosis, the skin cancers will be named as so and the remaining ones will be called skin diseases

Diagnostic	Nº of samples	% biopsied
Squamous Cell Carcinoma (SCC)	192	100%
Basal Cell Carcinoma (BCC)	845	100%
Melanoma (MEL)	52	100%
Actinic Keratosis (ACK)	730	24.4%
Nevus (NEV)	244	24.6%
Seborrheic Keratosis (SEK)	235	6.4%
Total	**2298**	**58.4%**

Table 12 – The number of samples for each type of skin disorder present in the PAD-UFES-20 dataset. Observe we have three types of skin cancer (BCC, SCC, and MEL) and three types of skin diseases (ACK, NEV, and SEK).

requested for clinical diagnostic confirmation. In this dataset, all BCC, SCC, and MEL are proved by biopsy. The histopathology procedure involves the following steps: 1) the collection of a skin fragment, 2) tissue fixation in formaldehyde (at a concentration of 10%), 3) macroscopic analysis of the skin fragment, 4) histological processing, 5) producing the microscope slides, 6) and a microscopic study with diagnosis' formulation and interpretation (Werner, 2009). As described in Table 12, 58.4% of the skin lesions in PAD-UFES-20 dataset are proved by biopsy, including 100% of the skin cancer. This number is compatible with other skin lesion datasets described in literature (Tschandl; Rosendahl; Kittler, 2018; Combalia et al., 2019).

Regarding the metadata features, they were collected according to the anamnesis of a patient, which beyond the skin lesion screening, dermatologists also consider the anatomical region, diameter, ulceration, itching, bleeding, among others characteristics of the skin lesion (Wolff et al., 2017; Azulay, 2017). In addition, risk factors are also taken into account such as exposure to chemicals, cancer history, and the type of skin (Duarte et al., 2018). In the dataset there are approximately 120 different anatomical regions used by the PAD's dermatologists and pathologists. We clustered these regions in 15 macro-regions that are more frequent and have more potential to raise a skin lesion, they are: face, scalp, nose, lips, ears, neck, chest, abdomen, back, arm, forearm, hand, thigh, shin, and foot. As skin lesions have preferences for some regions of the body (Wolff et al., 2017; Azulay, 2017), it is an important feature to consider.

The last step of the collection workflow is quality selection. The goal of this step is to review the patient clinical data and remove poor quality images. All data from the appointment are filled by senior medical students and the images are collected using different types of smartphone devices. In addition, the smartphone application allows the user to crop the image to select only the region of interest. As a result, images in different conditions are uploaded to the server. Thereby, during the quality selection, we delete those images according to the following rules:

- The image resolution is very poor and it is not possible to identify the lesion.
- The patient may be identified because of a tattoo, for example.
- The lesion is completely occluded by hair or ink marking.

It is worth noticing that the images present in the dataset have different resolutions, sizes, and lighting conditions. Essentially, an application to detect skin cancer using clinical images needs to deal with such variability. Thus, it aims to close to the real world.

We also review all patient clinical data in order to correct typos and information that are clearly wrong, for example, birth dates before 1900 or skin lesions' diameter that are much bigger than it looks on the image according to visual inspection. For these cases, we re-checked the physical files in order to fix the information. If it is not possible to fix it, we remove the wrong information from the clinical data and it becomes missing data. To conclude, for those cases in which the pathology yielded a biopsy that is inconclusive, we also removed the sample from the dataset.

It is important to mention the dataset was collected along with the Dermatological and Surgical Assistance Program (PAD) of the Federal University of Espírito Santo. The program is managed by the Department of Specialized Medicine and was approved by the university ethics committee (nº 500002/478) and the Brazilian government through Plataforma Brasil (nº 4.007.097), the Brazilian agency responsible for research involving human beings. In addition, all data were collected under patient consent and the patient's privacy is completely preserved.

5.1.2 Data description

All samples within PAD-UFES-20 represents a skin lesion of a patient that is composed of an image and a set of metadata. A patient may have one or more skin lesions and a skin lesion may have one or more images. In total, there are 1,373 patients, 1,641 skin lesions, and 2,298 images present in the dataset. As previously mentioned, the images present in the dataset have different sizes because they are collected using different smartphone devices. In Figure 33 is illustrated image samples for each skin lesion present in the dataset.

The metadata associated with each skin lesion is composed of up to 21 features. We describe all of them in the following:

- **smoke**: a boolean to map if the patient smokes cigarettes.
- **drink**: a boolean to map if the patient consumes alcoholic beverages.

(a) BCC (b) SCC (c) MEL (d) ACK (e) NEV (f) SEK

Figure 33 – Samples of each type of skin lesion present in PAD-UFES-20 dataset. SCC, BCC, and MEL are skin cancers and NEV, SEK and ACK are skin diseases.

- **background of the father** and **background of the mother**: a string representing the country in which the patient's father and mother descends.
- **age**: an integer representing the patient's age.
- **pesticide**: a boolean to map if the patient uses pesticides – since most of them are work farmers.
- **gender**: a string representing the patient's gender.
- **skin cancer history**: a boolean to map if the patient or someone in their family has had skin cancer in the past.
- **cancer history**: a boolean to map if the patient or someone in their family has had any type of cancer in the past.
- **piped water**: a boolean to map if the patient has access to piped water in their home.
- **sewage system**: a boolean to map if the patient has access to a sewage system in their home.
- **fitspatrick**: a integer representing the Fitspatrick skin type (Wolff et al., 2017).
- **anatomical region**: a string representing one of the 15 macro-regions previously described.
- **diameter 1** and **diameter 2**: a float representing the skin lesions' horizontal and vertical diameters.
- **itch**: a boolean to map if the skin lesion itches.
- **grew**: a boolean to map if the skin lesion has recently grown.
- **hurt**: a boolean to map if the skin lesion hurts.
- **changed**: a boolean to map if the skin lesion has recently changed.

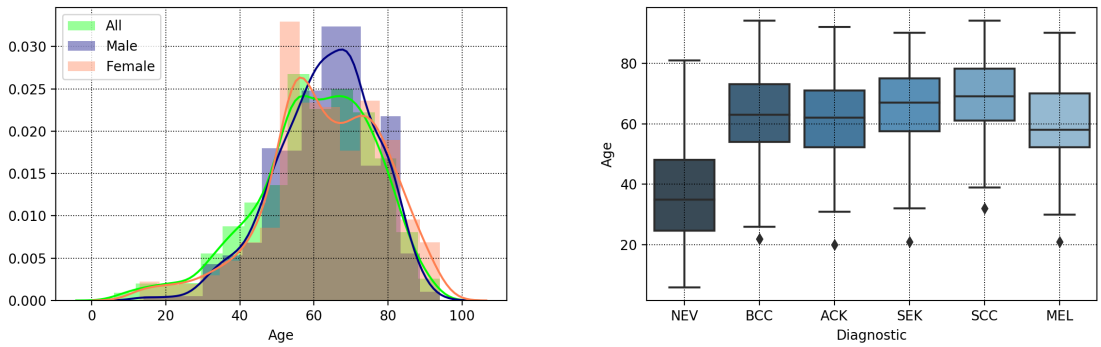
- **bleed**: a boolean to map if the skin lesion has bled.
- **elevation**: a boolean to map if the skin lesion has an elevation.

It is important to note that some features may be missing for some lesions because they depend on the patient's answers and if the lesion was biopsied or not. In general, missing values are left blank. When a patient does not know the answer for some question – for example, he/she does not know their father's background – we fill the feature as UNK (unknown). All data records are available on Mendeley Data (Pacheco et al., 2020).

5.1.3 Clinical features analysis

In order to get a gist of patient clinical information on skin cancer detection in PAD-UFES-20 dataset, we performed an Exploratory Data Analysis (EDA) using the clinical features described in the previous section. The complete analysis is available on the thesis repository on Github[2]. In this section, we highlight the main findings.

We start analyzing the patient's age distribution. In Figure 34 is depicted the age histogram and boxplots stratified by gender and diagnostic, respectively. As we can note, there is no significant difference between female and male age distribution. In general, the patient age average is around 62 years old. Observing the boxplots in Figure 34, we can see that the patients' age may be useful to distinguish NEV from the other skin lesions since it presents a lower median.



(a) Histogram per gender

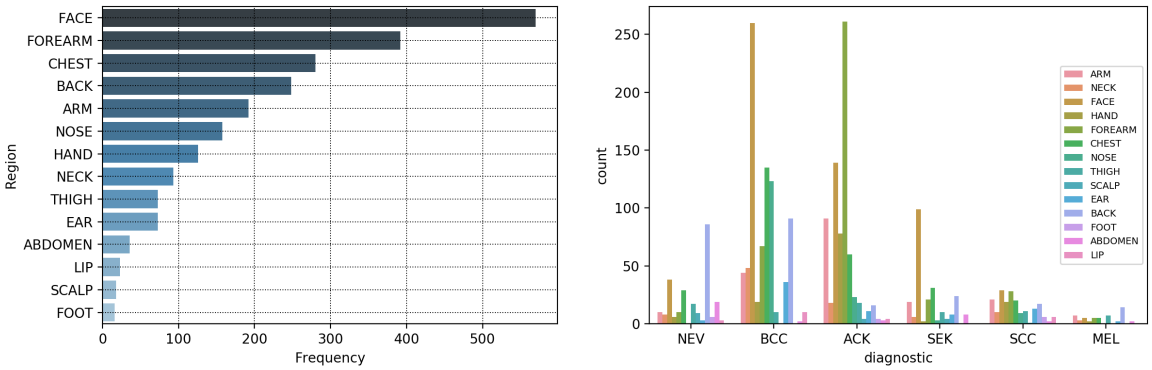
(b) Boxplots per diagnostic

Figure 34 – The patients' age histogram stratified by gender and boxplots per diagnostic.

In terms of anatomical region, in Figure 35 is shown the total frequency of each region and the frequency per diagnostic. As we can note, the most common anatomical regions are face and forearm. It is expected since sun exposure is the main risk factor for skin cancer and normally these regions are not covered by clothes in daily life. In

[2] <https://github.com/paaatcha/my-thesis/blob/master/benchmarks/pad/analysis/pad-ufes-20-analysis.ipynb>

addition, ACK and BCC are more common in these regions and they are the most frequent diagnostics in PAD-UFES-20. We also analyzed the diameters per diagnostic. As shown in Figure 36, the skin lesion diameters have similar size, which suggests that this attribute is not useful to distinguish them.



(a) Histogram per gender

(b) Boxplots per diagnostic

Figure 35 – The total frequency of each region and the frequency per diagnostic.

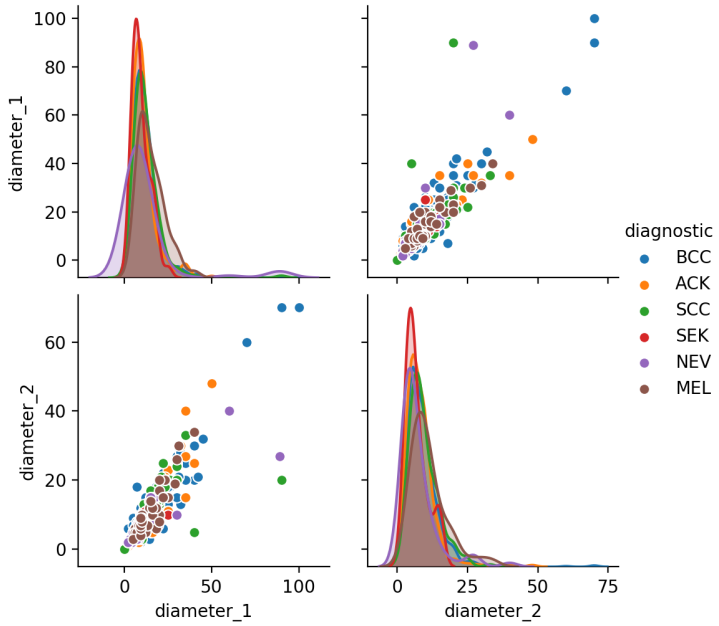
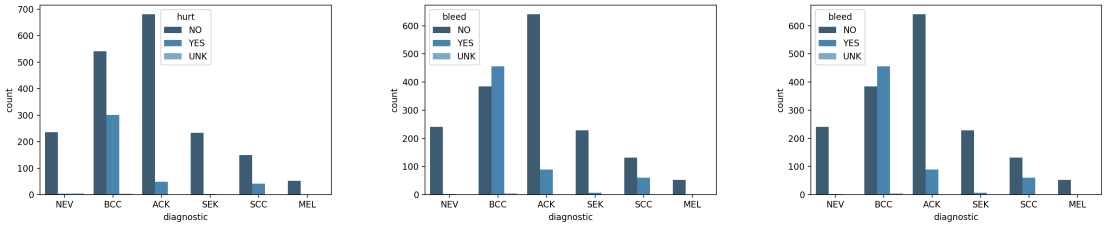


Figure 36 – The skin lesion diameters distribution and scatterplot stratified per diagnostic.

In Figure 37 is presented the bar plots for three boolean features that provide important insights about this dataset. As described in the previous section, these three features are acquired through questioning the patient. We observe that pigmented skin lesions usually do not hurt, bleed, and itch more than the non-pigmented ones. Thereby, these features seem to be quite useful to differentiate these lesions.

We also analyzed the remaining features present in the dataset. In general, we summarize the key points as follows:



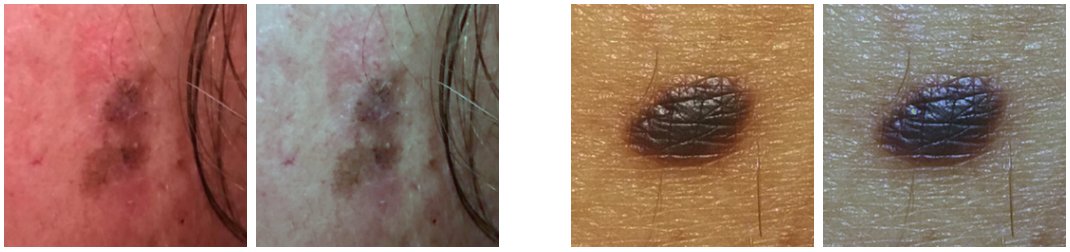
(a) Does the skin lesion hurts? (b) Has the skin lesion bleed? (c) Does the skin lesion itch?

Figure 37 – Bar plots for three features based on the questions that dermatologists make to the patients.

- It is expected that these features improve the model performance for pigmented and non-pigmented lesions detection.
- Certain features, such as a change in the lesion pattern and elevation are important for MEL detection.
- In general, SCC and BCC share the same clinical features values. Both bleed, hurt, itch, present elevation, occur in the same age range and have the same preferred region. Thus, it is expected that these features will not be helpful in distinguishing between SCC and BCC.
- Germany and Pomerania are the most common answer for father/mother background. In fact, it is expected since most of the people from the 11 cities in which PAD takes place descends from immigrants of these countries. Nonetheless, it might not be useful to distinguish skin lesions.
- Most of the patients reported they do not drink or smoke. Thus, we can not find a correlation between skin lesion and this risk factor in PAD-UFES-20 dataset.
- The gender distribution is quite close for all skin lesions, which also makes it hard to find correlation.

5.2 Experimental results

In this section, we present experiments using the dataset described in Section 5.1. We perform the proposed methods described in Chapter 3 and 4 to detect skin cancer using clinical images and patient demographics. We start by describing the experiments setup; next, we carry out experiments regarding the meta-data and images combinations; Then, we create an ensemble of deep models and evaluate the aggregation methods presented in Chapter 4; Finally, we provide a discussion about the case study.



(a) Original (b) Processed (c) Original (d) Processed

Figure 38 – The difference between the original images (left) and the images after the color constancy pre-processing (right).

5.2.1 Experiments setup

The experiments setup is quite similar to the previous experiments carried out in Section 3.4 and 4.3. We perform all experiments using five competitive CNN architectures: EfficientNet-B1 (Tan; Le, 2019), DenseNet-121 (Huang et al., 2017b), MobileNet-v2 (Sandler et al., 2018), ResNet-50 (He et al., 2016), and VGGNet-13 (Simonyan; Zisserman, 2014). Likewise the previous sections, all models are trained using their original architecture and including the baseline concatenation method, the MetaBlock, and MetaNet. The training phase is carried out using the same configuration described in Sections 3.4.1 and 4.3, i.e., using SGD optimizer along with weight decay, momentum, and early stopping.

For all experiments, we also reserved approximately 10% of each dataset for testing respecting the labels' frequency distribution. The remaining 90% of data are used on the training phase through 5-fold cross-validation for assessing the effectiveness of the models. All methods performance are also measured computing the average and standard deviation of the accuracy (ACC), balanced accuracy (BACC), aggregated area under the curve (AUC), and the cross-entropy (Loss). In order to compare the experiments results, we also perform the Friedman test followed by the Wilcoxon test (if applicable), using $p_{\text{value}} = 0.05$, and the A-TOPSIS ranking algorithm (Krohling; Pacheco, 2015).

The PAD-UFES-20 dataset presents similar characteristics as any medical dataset. As we describe in the previous sections, the amount of data is not large and the dataset is imbalanced. In addition, as the dataset is collected using smartphone cameras, it presents fewer details of the skin lesion when compared to dermoscopic images. Also, camera resolution and illumination affect the images' quality. Vasconcelos and Vasconcelos (2017) presented a review in which they discuss approaches to handle these types of issues in skin cancer datasets. In summary, they state we may tackle these problems with transfer learning, up/down-sampling, data augmentation, and using an ensemble of models. Additionally, Barata, Celebi and Marques (2014) showed the benefits of using color constancy algorithms for skin cancer detection. Following their recommendations, we applied the shades of gray method (Finlayson; Trezzi, 2004) for all images before the training phase. The difference between the clinical images with and without the color

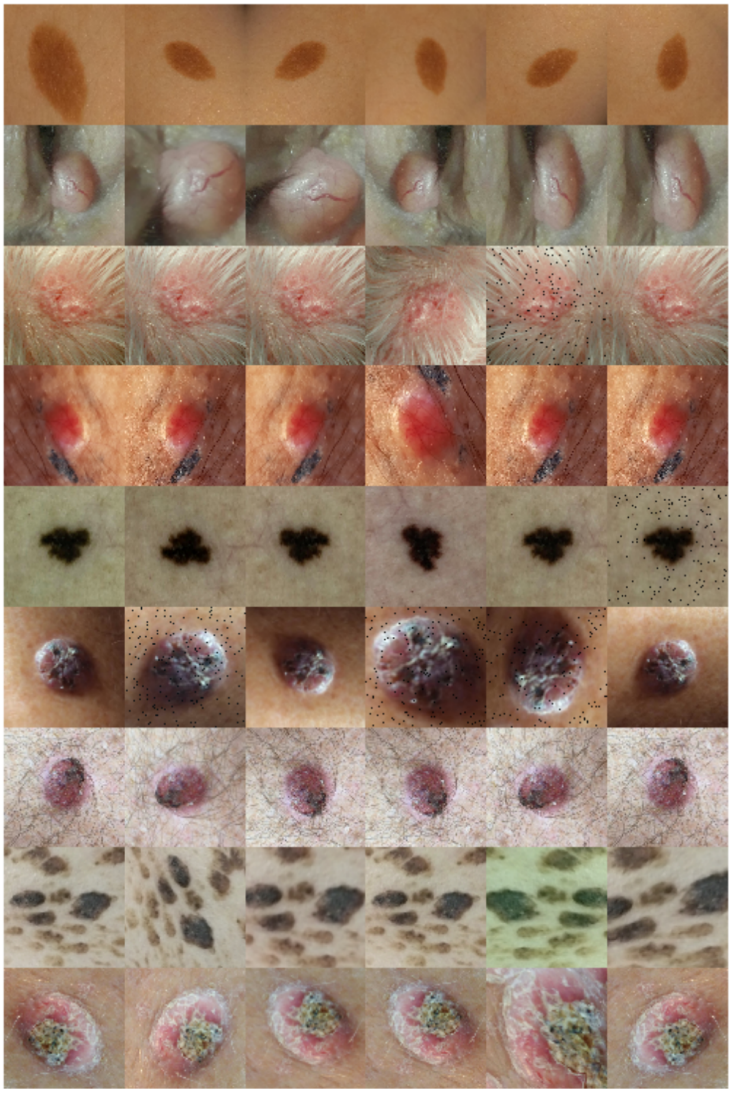


Figure 39 – The result of the data augmentation employed in this experiment for some samples from PAD-UFES-20 dataset.

constancy pre-processing can be seen in the samples depicted in Figure 38.

We also applied data augmentation using the following image processing operations: we adjust brightness, saturation, and hue. We also apply horizontal and vertical rotations, translations, re-scale, shear, random noise, and blur. In Figure 39 is shown the result of these data augmentation operations for some random samples from PAD-UFES-20 dataset.

To tackle the labels' imbalanced issue, we use a weighted loss function based on the label's frequency. The weight for a given label l is computed as follows:

$$w_l = \frac{N}{n_l} \tag{5.1}$$

where N is the total number of samples in the dataset and n_l is the number of samples for label l. We also tried to use oversampling to equalize the number of samples for each class. However, it created a high bias for the MEL label, which resulted in a lower performance

compared to the weighted loss function approach.

Regarding the clinical features, likewise in Section 3.4 we also applied the one-hot strategy to encode the features represented by strings for the same reasons we previously describe, i.e., most of data are boolean and this strategy can properly encode them. Thus, the 21 attributes are mapped in a array of 81 values. These features are used as meta-data to support the skin lesion classification.

5.2.2 Experiment results

We start the case study experiments by evaluating the methods to combine images and meta-data proposed in Chapter 3. In this case study, each sample in the dataset is a skin lesion that is represented by an image and clinical data, which is interpreted as meta-data. We perform the classification considering only the images, combining images and meta-data using the concatenation approach, MetaNet, and MetaBlock. For the concatenation approach, we use the combination factor c_f equal to 0.8, which results in 324 neurons in the feature learning block – see equation 3.6. In Appendix B.1, we present a sensitivity analysis regarding the combination factor parameter.

In this part of the experiment, we compare the deep learning models' performance with and without considering the meta-data. In Tables 13, 14, 16, and 15 are presented the results, in terms of mean and standard deviation, for each method. To ease the visualization, in Table 17 is summarized the performance only in terms of BACC for each method. As we can note from the tables, there is a notable improvement for all metrics when the meta-data is included into the classification process. Quantitatively, in terms of BACC, there is an average improvement of around 9% when meta-data are used. Comparing with the results achieved for ISIC 2019 dataset, presented in Section 3.4, the improvement was much higher, in particular, because we have access to a more valuable 21 patient clinical attributes instead of only 3.

No meta-data				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B4	0.656 ± 0.027	0.64 ± 0.029	0.911 ± 0.006	0.867 ± 0.053
DenseNet-121	0.636 ± 0.055	0.64 ± 0.042	0.893 ± 0.02	0.98 ± 0.117
MobileNet-V2	0.655 ± 0.016	0.637 ± 0.018	0.898 ± 0.01	0.934 ± 0.03
ResNet-50	0.616 ± 0.051	0.651 ± 0.05	0.901 ± 0.007	1.0 ± 0.072
VGGNet-13	0.709 ± 0.019	0.654 ± 0.022	0.901 ± 0.003	0.865 ± 0.024

Table 13 – Deep learning models' performance for PAD dataset considering only clinical images

Observing the models' performance when the meta-data is employed, we note that they present performances varying from 71.7% to 77% in terms of BACC. Similar to ISIC 2019 dataset, the MetaBlock presents a more stable performance compared

Concatenation				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B4	0.765 ± 0.032	0.758 ± 0.012	0.945 ± 0.005	0.637 ± 0.053
DenseNet-121	0.742 ± 0.035	0.747 ± 0.019	0.932 ± 0.005	0.786 ± 0.057
MobileNet-V2	0.738 ± 0.026	0.741 ± 0.017	0.927 ± 0.007	0.767 ± 0.087
ResNet-50	0.741 ± 0.014	0.728 ± 0.029	0.929 ± 0.006	0.833 ± 0.08
VGGNet-13	0.712 ± 0.035	0.72 ± 0.018	0.929 ± 0.002	0.765 ± 0.051

Table 14 – Deep learning models' performance for PAD dataset considering clinical images and patient demographics. In this case, the concatenation method is applied to combine both data with a combination factor set to 0.8.

MetaBlock				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B4	0.748 ± 0.018	0.77 ± 0.016	0.944 ± 0.004	0.636 ± 0.031
DenseNet-121	0.723 ± 0.037	0.746 ± 0.039	0.931 ± 0.006	0.743 ± 0.047
MobileNet-V2	0.724 ± 0.016	0.754 ± 0.014	0.938 ± 0.005	0.701 ± 0.037
ResNet-50	0.735 ± 0.013	0.765 ± 0.017	0.935 ± 0.004	0.717 ± 0.034
VGGNet-13	0.728 ± 0.022	0.736 ± 0.018	0.933 ± 0.002	0.781 ± 0.03

Table 15 – Deep learning models' performance for PAD dataset considering clinical images and patient demographics. In this case, the MetaBlock is applied to combine both data.

MetaNet				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-B4	0.744 ± 0.014	0.737 ± 0.017	0.931 ± 0.007	0.718 ± 0.041
Densenet-121	0.745 ± 0.022	0.745 ± 0.029	0.932 ± 0.008	0.723 ± 0.063
MobileNet-V2	0.700 ± 0.013	0.717 ± 0.020	0.922 ± 0.005	0.787 ± 0.027
ResNet-50	0.732 ± 0.054	0.742 ± 0.019	0.936 ± 0.006	0.707 ± 0.048
VGGNet-13	0.749 ± 0.037	0.754 ± 0.033	0.937 ± 0.011	0.706 ± 0.073

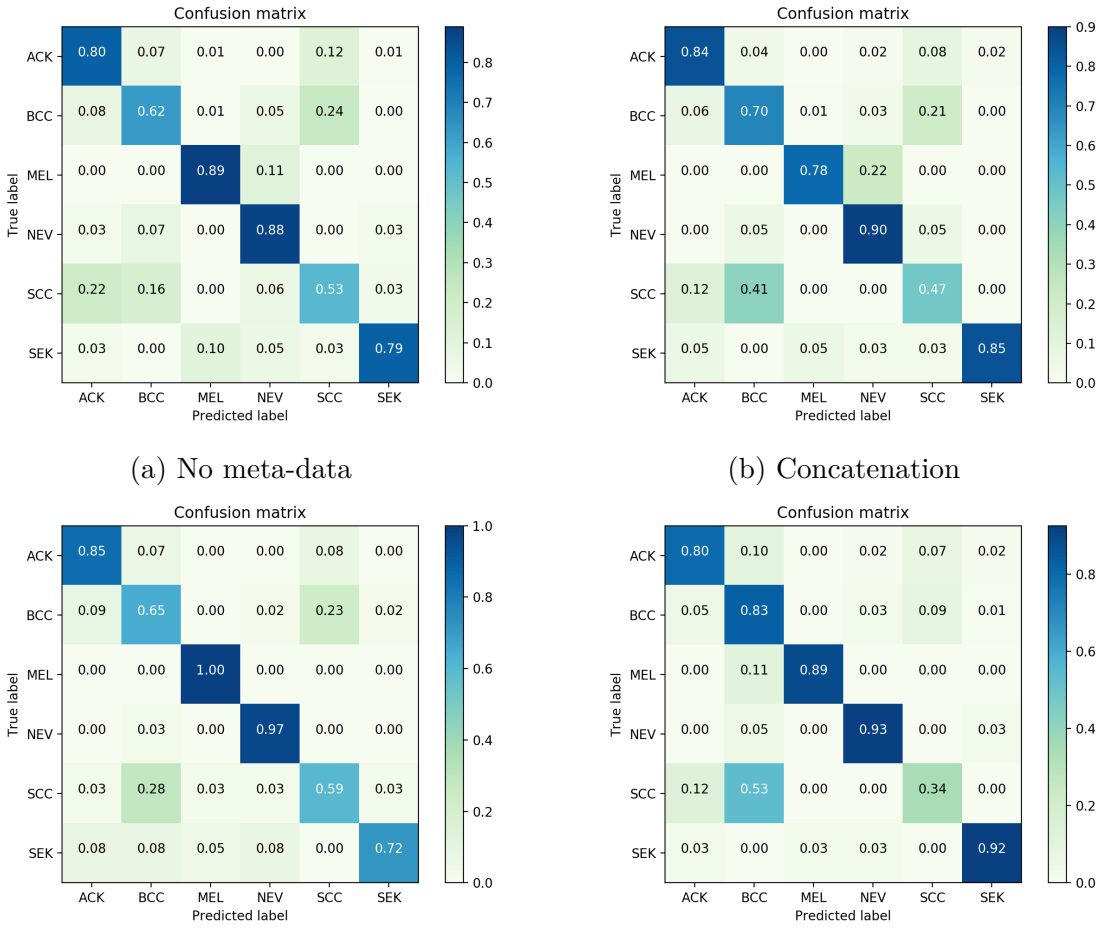
Table 16 – Deep learning models' performance for PAD dataset considering clinical images and patient demographics. In this case, the MetaNet is applied to combine both data.

to the concatenation and MetaNet. In this experiment, we selected the ResNet-50 for further analysis. In Figure 40 and 41 are depicted the confusion matrix and ROC curve, respectively, for this model considering all methods[3]. These plots show an interesting result, in general, the meta-data helped to improve the diagnostic rates for ACK, MEL, NEV, and SEK. However, it increases miss-classification between SCC and BCC. This result is in accordance with the analysis provided in Section 5.1.3, in which we show that both lesions share almost the same values of clinical features. In fact, even dermatologists get confusing regarding these two lesions. As shown in Figures 33a and 33b, both lesions are very similar and distinguishing them is a challenging task even for experts using a dermatoscope. Nonetheless, confusing SCC and BCC is not quite a problem, since both are skin cancer and need to be biopsied. The real problem is confusing them with ACK,

[3] The plots for the remaining models are available in Appendix B.5

Model	No meta-data	Concatenation	MetaBlock	MetaNet
EfficientNet-B4	0.64 ± 0.029	0.758 ± 0.012	$\mathbf{0.77 \pm 0.016}$	0.737 ± 0.017
DenseNet-121	0.64 ± 0.042	$\mathbf{0.747 \pm 0.019}$	0.746 ± 0.039	0.745 ± 0.029
MobileNet-V2	0.637 ± 0.018	0.741 ± 0.017	$\mathbf{0.754 \pm 0.014}$	0.717 ± 0.020
ResNet-50	0.651 ± 0.05	0.728 ± 0.029	$\mathbf{0.765 \pm 0.017}$	0.742 ± 0.019
VGGNet-13	0.654 ± 0.022	0.72 ± 0.018	0.736 ± 0.018	$\mathbf{0.754 \pm 0.033}$

Table 17 – Comparing the models' performance in terms of BACC for each method. In bold is highlighted the highest average BACC for each model.



(a) No meta-data

(b) Concatenation

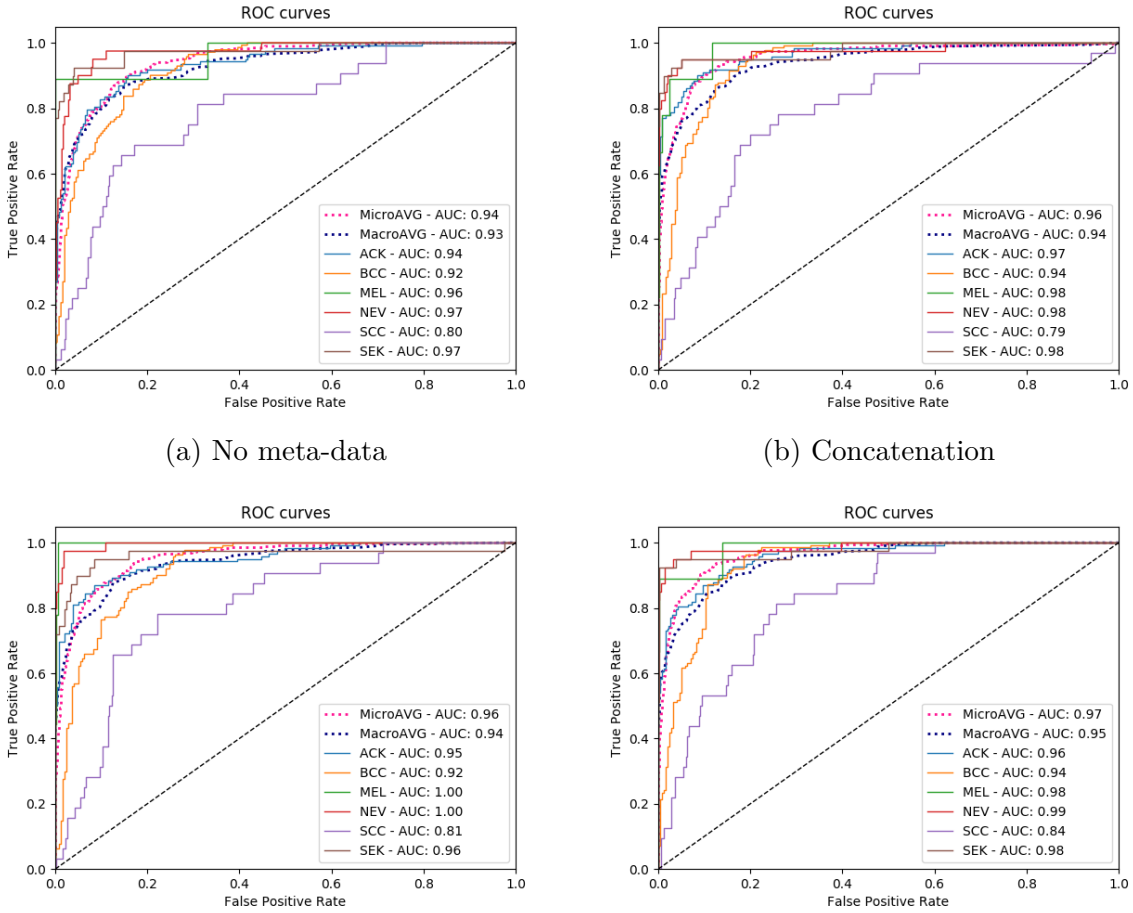
(c) MetaBlock

(d) MetaNet

Figure 40 – Confusion matrices for ResNet-50 considering all methods.

which is just a minor skin disease that is treated without a surgical process. For MEL, the deadliest case of skin cancer, the MetaBlock approach achieves maximum performance, while the concatenation and MetaNet do no present an improvement.

We performed the Friedman and Wilcoxon tests, considering the BACC metric, to compare the performance of the method. The Friedman test returned $p_{\text{value}} \approx 5 \times 10^{-11}$. Thus, we performed the Wilcoxon test, in which the p values are presented in Table 18. As we can see, the test returns $p_{\text{value}} < 0.05$ for all pairwise comparisons, except the pair MetaNet-Concatenation. In other words, it means all methods are statistically different



(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 41 – ROC curves for ResNet-50 considering all methods and all skin lesions.

from each other, except the MetaNet and Concatenation approaches. Based on the results presented throughout this section, we may conclude that the MetaBlock performs better than the concatenation and MetaNet approaches. This assumption is confirmed by the A-TOPSIS rank, depicted in Figure 42, which shows the MetaBlock with a higher rank score while the concatenation and MetaNet have quite close values.

Pair	$p_{\textbf{value}}$
No meta-data - Concatenation	$\sim 10^{-5}$
No meta-data - MetaBlock	$\sim 10^{-5}$
No meta-data - MetaNet	$\sim 10^{-5}$
MetaBlock - Concatenation	0.008
MetaBlock - MetaNet	0.04
MetaNet - Concatenation	0.81

Table 18 – The result of the Wilcoxon pairwise test for all methods.

In order to visualize the contribution of the patient demographics in distinguishing the skin lesion in this dataset, we applied the t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten; Hinton, 2008) to reduce the ResNet-50 features into low-dimensional

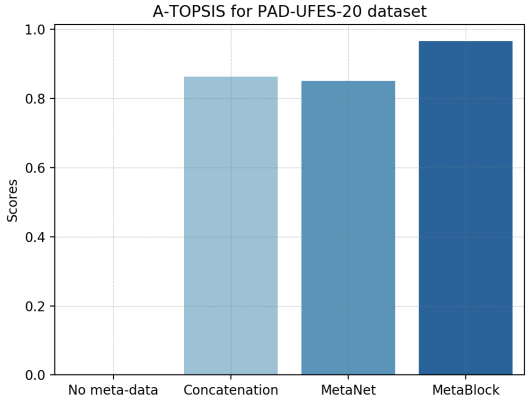


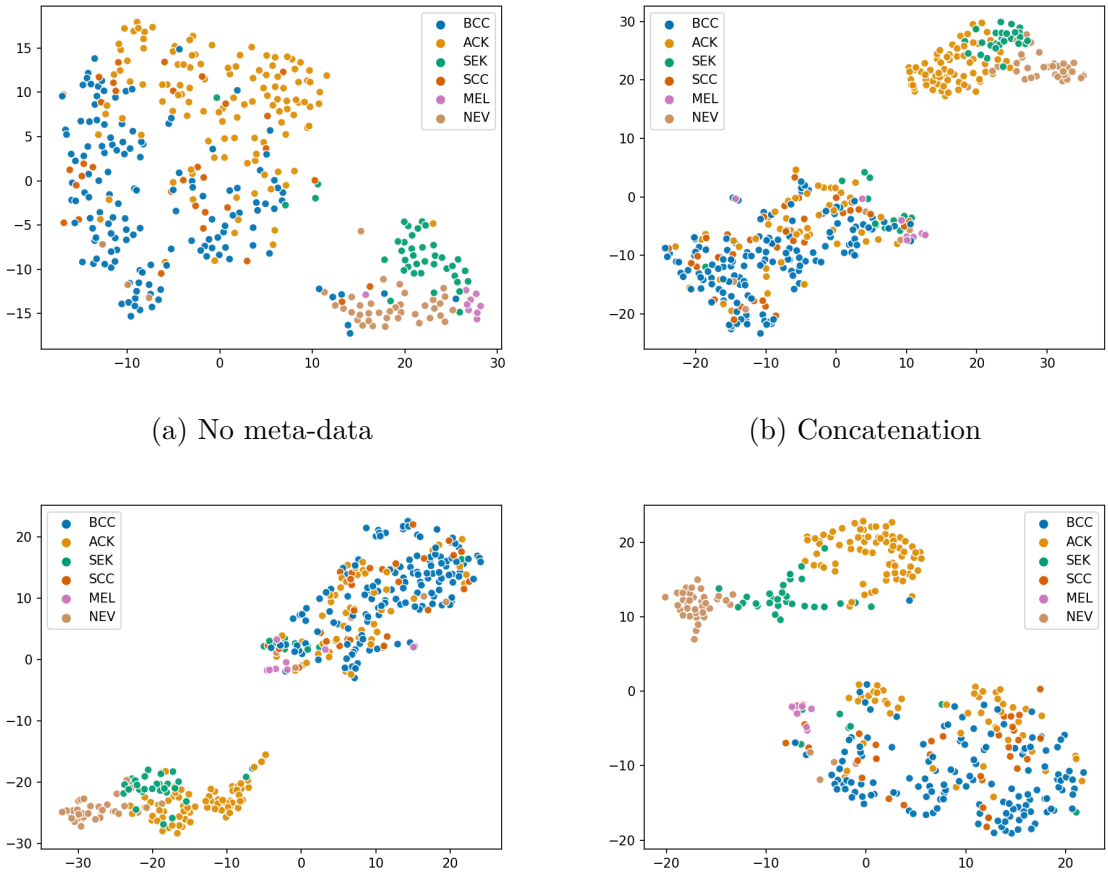
Figure 42 – The A-TOPSIS rank for the three methods considering the BACC metric

space of two dimensions. The scatter plot of this reduction is shown in Figure 43. As we can see, the patient demographics impacted more in distinguishing the pigmented skin lesions (NEV, MEL, SEK) from the non-pigmented ones (BCC, SCC, and ACK). This result is in line with our previous analysis in Section 5.1.3. BCC and SCC samples are still hard to cluster, which also reflected in the confusion matrices plot in Figure 40.

In addition to the t-SNE scatter plots, in Figure 44 is depicted examples of miss-predictions assigned by the ResNet-50 without using clinical data that were correctly classified by the model when the MetaBlock is used to combine the clinical images and the meta-data. As we can note, there are two cases of cancer that were predicted as skin diseases, i.e., a BCC was predicted as NEV and a MEL was predicted as SEK. These false negatives are the worst scenario for skin cancer detection, in particular, for the melanoma case. This example shows the importance to propose methodologies to properly combine the clinical images and patient clinical information. For these cases, the clinical data were fundamental to properly predict the dysplasia.

To conclude this case study, we use the results of the combination methods to create an ensemble of the five models previously presented. As described in Chapter 4, many real-world problems apply an ensemble in order to improve performance and increase robustness. We have the same goal in this part of the case study. The experiment in this section has the same structure that the one carried out in Section 4.3. In Table 19 is presented the performance of the seven aggregation approaches for the ensemble of models considering the clinical images and patient demographics. To ease the comparison, we also include the performance of the EfficientNet-B4, the model that achieves the highest BACC in the ensemble.

All seven aggregation methods present similar performance for all metrics. Quantitatively, compared to the best model within the ensemble, the average ACC and BACC were improved in almost 1.2% and 1%, respectively; the Loss was slightly reduced by the



(a) No meta-data

(b) Concatenation

(c) MetaNet

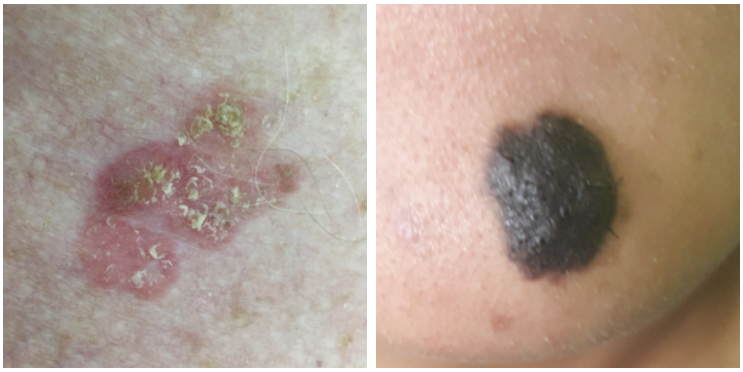
(d) MetaBlock

Figure 43 – The t-SNE visualization considering the ResNet-50 model with and without using the meta-data

Method	ACC	BACC	AUC	Loss
AVG	0.758 ± 0.016	0.778 ± 0.013	$\mathbf{0.944 \pm 0.004}$	0.647 ± 0.022
MAX	0.752 ± 0.024	0.775 ± 0.022	0.942 ± 0.003	0.692 ± 0.023
PROD	$\mathbf{0.760 \pm 0.020}$	0.777 ± 0.022	0.943 ± 0.003	1.671 ± 0.138
MV	0.755 ± 0.023	0.775 ± 0.015	0.910 ± 0.007	4.623 ± 0.494
SWA	0.759 ± 0.017	0.778 ± 0.012	0.934 ± 0.004	0.678 ± 0.044
NP-AVG	0.759 ± 0.017	0.778 ± 0.013	0.942 ± 0.004	0.645 ± 0.022
LewDir	0.752 ± 0.021	0.779 ± 0.010	0.942 ± 0.003	$\mathbf{0.622 \pm 0.024}$
EfficientNet-B4	0.748 ± 0.018	0.770 ± 0.016	0.944 ± 0.004	0.636 ± 0.031

Table 19 – Performance achieved by each aggregation approach for ensemble of CNN models trained on PAD-UFES-20 dataset.

(a) BCC predicted as NEV (b) MEL predicted as SEK



(c) SCC predicted as BCC (d) SEK predicted as NEV

Figure 44 – Examples of skin lesion wrongly predicted by the ResNet-50 model without using the meta-data that was corrected by the MetaBlock approach.

LewDir approach; and the AUC is basically the same, except for MV, which reflects in the ROC curve depicted in Figure 45. We performed the Friedman test using the BACC metric as a reference, and as expected the method returned $p_{\text{value}} > 0.05$, which indicates that there is no aggregation method statistically different for this dataset. This result is in line with the A-TOPSIS rank depicted in Figure 46.

We also applied the online ensemble selection based on the weights assigned by our method. In Table 20 is presented the ensemble pruning that selects from 1 to 4 models from the group. In the last line, the results for all models are included to ease comparison. Recall that as the weights change according to the sample, the models selected may change every time the ensemble evaluated a new sample. As we can see, the results obtained for PAD-UFES-20 dataset are similar to the ones obtained for ISIC in Section 4.3. The performance in terms of average BACC are similar to all number of models, however, the best result is still achieved when the ensemble contains all models. As the number of models decreases, the loss function decreases. For example, it is possible to get similar ACC by decreasing the number of models in the ensemble; however, the BACC and the prediction confidence decrease. Essentially, this experiment suggests that LewDir is properly assigning the weights of the models, i.e., the weak models are already getting low weights and does not affect too much in the final prediction.

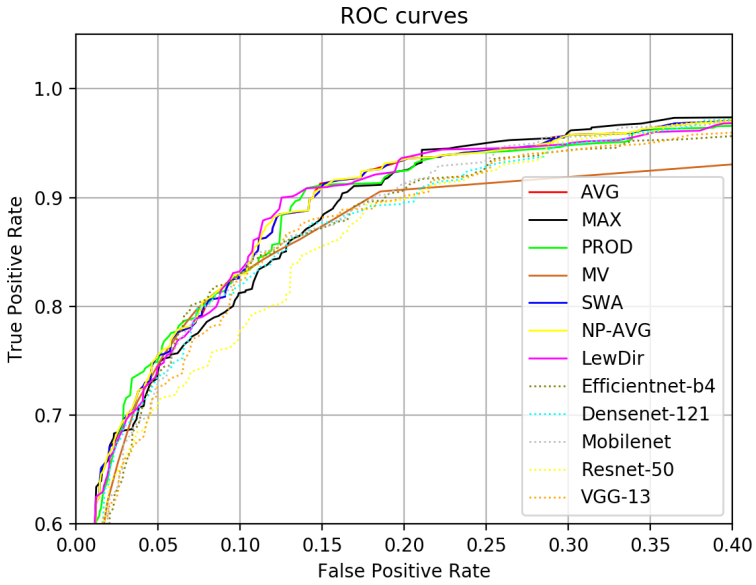


Figure 45 – The macro average ROC for the ensemble of CNN models considering the same folder for all methods.

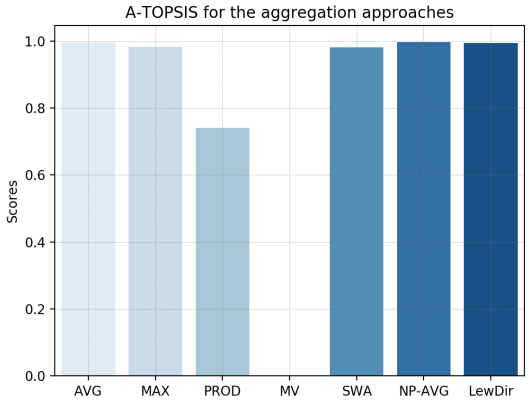


Figure 46 – The A-TOPSIS rank for all aggreationg methods and CNN models.

Nº of models	**ACC**	**BACC**	**AUC**	**Loss**
1	0.737 ± 0.031	0.759 ± 0.030	0.932 ± 0.004	0.762 ± 0.035
2	0.749 ± 0.019	0.775 ± 0.019	0.941 ± 0.003	0.692 ± 0.038
3	0.745 ± 0.030	0.767 ± 0.028	0.941 ± 0.004	0.674 ± 0.032
4	0.750 ± 0.023	0.772 ± 0.017	0.941 ± 0.003	0.666 ± 0.028
All	0.752 ± 0.021	0.779 ± 0.020	0.942 ± 0.003	0.622 ± 0.0024

Table 20 – Ensemble selection according to the weights assigned by our proposed method.

5.3 Discussion

The results presented in this case study confirm the hypothesis raised by Brinker et al. (2018) that patient clinical data are important to improve deep learning performance for skin cancer detection. Both proposed approaches, the concatenation and MetaBlock, to combine the images and meta-data presented higher performance than the models without using the meta-data. The results achieved in this experiment are competitive to other state-of-the-art deep learning approaches reported in the literature. First of all, as we showed, the concatenation approach achieves similar performance to MetaNet (Li et al., 2020) and the MetaBlock generally performs better than both previous methods. In addition, Han et al. (2018) applied a ResNet-152, which is similar to ResNet-50, to classify 12 skin lesions considering two different datasets. They reported an average AUC of 0.91 and 0.89 for each dataset, respectively. Esteva et al. (2017) validated a GoogleNet model for benign/malignant classification. They achieved an AUC equal to 0.96, which is on par with the dermatologist level. These two works also use clinical images; however, they do not consider the patient demographics. Kharazmi et al. (2018) took into account the clinical information, but for dermoscopic images. The authors proposed an approach based on sparse autoencoder (SAE) (Ng et al., 2011) to combine clinical features and dermoscopic images. They achieved an improvement in AUC from 0.847 to 0.911, when the clinical data were included. Lastly, Udrea et al. (2020) applied a proprietary algorithm[4] to classify clinical images into low or high-risk skin lesions. They reported a sensitivity and specificity of 95.1% and 78.3%, respectively, to detect (pre)malignant conditions. Even though only one of these works consider clinical images and patient demographics, the results achieved in this study case are in accordance with them. In addition, these results are also similar to the ones achieved by the PAD dermatologists[5], which is our baseline for this case study. However, it is quite important to note that this study case considers only the six most common skin lesions at the PAD. The PAD dermatologist must deal with more than 50 different kinds of skin lesions during the appointments.

The ensemble of deep models presented a slight gain of performance when compared to the models individually. The proposed method achieved a BACC approximately 1% higher than the best model in the ensemble. However, the performance of the remaining methods are quite close to the proposed one, which suggests there is a room for improvement of LewDir. In fact, in the conclusion section, we indicate a path we believe that will lead to improvements in the proposed method. In general, as Vasconcelos and Vasconcelos (2017) reviewed, an ensemble of models is an effective way to improve performance and robustness for skin cancer detection. However, there is no silver bullet. As stated before, employing an ensemble is a trade-off between performance and computational resources.

[4] This algorithm belong to Skin Vision Inc and it is not publicly.

[5] As the PAD dermatologists are not anonymized, we understand that is unethical to reveal their detection performance and reserved this information only for the PAD staff.

For skin cancer detection, if there is enough hardware available, an ensemble of deep models may be useful to improve the prediction certainty, in particular for melanoma. As shown in the experiments, the ensemble can provide a boost in terms of performance, however, developing approaches to combine the clinical images and patient demographics was demonstrated to impact much more in the skin cancer detection.

The long term goal of the PAD researchers is to embed the proposed model into a smartphone application in order to assist them to detect skin cancer during the appointments. Actually, we have already developed a prototype (Castro et al., 2020); however, it is necessary to perform exhaustive tests[6] before deploying such a system. Regarding this matter, we believe the AI researcher community must be careful in providing auto-diagnostic solutions to general patients. Beyond problems regarding patient confidentiality and privacy, such applications have the potential to harm or mislead patients with an incorrect diagnostic. Let us consider a hypothetical situation of a false negative for melanoma to a given user. It may delay their treatment and, in the worst scenario, it may lead them to death. This is a serious problem that we need to confront. We understand that the best way to use these solutions is along with experts. In other words, a CAD system should be used to assist detection, and an expert must have the final word.

To conclude, it is important to note that the PAD-UFES-20 dataset is composed of clinical and biopsied diagnostics. As described in Section 5.1, 58.4% of all samples are biopsy-proven, including 100% of the skin cancer. In fact, having a mix of diagnostic is a recurrent issue in skin cancer datasets. For example, consensus diagnosis is also applied to the dataset used by Esteva et al. (2017), Han et al. (2018), and in ISIC archive (ISIC, 2019). In order to reduce the number of unnecessary biopsies and consequently, the costs, most of the skin lesions are not referred to histopathology. In this context, the achieved results represent an advance towards automated skin cancer detection, in particular to detectors using clinical images and patient demographics. However, it is still necessary to increase the dataset in terms of the number of samples and including more skin lesions. We expected that by using the software we developed in this work – see Appendix A –, in the next years this dataset may achieve the same number of samples and skin lesions as ISIC 2019.

[6] Tests were scheduled to be performed during this year, however, due to the COVID-19 pandemic, the PAD was deactivated for this year.

6 Conclusion

Despite the success of Deep Learning models for several relevant tasks, there are still many challenges in applying this technique, in particular for medical imaging analysis. In this thesis, we worked on two relevant topics involving Deep Learning: the combination of images and context data represented by meta-data and dynamic weighting an ensemble of deep models. For each topic, we summarize the key contributions as follows:

- We proposed two methodologies to combine images and meta-data. First, we described a concatenation approach, to work as a baseline, that is based on a combination factor between both source of data. Next, we introduced the Meta-data Processing Block (MetaBlock), a new algorithm that allows deep learning models to consider both meta-data and image in classification problems. In this method, the meta-data is used to support the classification by guiding the most relevant features extracted from the images without using feature concatenation.
- We proposed a new method that employs a Dirichlet distribution and the Mahalanobis distance to online determine the weights for an ensemble of deep models. Beyond reducing the impact of weak models on the aggregation operator, this method may also be used as an ensemble selection for each new sample evaluated by the set of models.

For the problem of combining images and context data, we performed experiments using different deep learning models architectures for the ISIC 2019 dataset to combine dermoscopy images and patient demographics. As we showed, the use of meta-data provided an average improvement of over 1% in balanced accuracy and reduced the loss for most models. Nonetheless, we also showed that the combination method is important for this task since the MetaBlock presented a better performance than the concatenation and MetaNet approaches. These latter approaches did not deal well with meta-data with just a few information, three patient attributes in this particular case. Nonetheless, it provided a fair improvement when the number of meta-data increases, as we showed in the case study.

For the online ensemble weighting problem, we performed experiments using four state-of-the-art medical imaging datasets. Results showed that the proposed method presented a competitive performance compared to the other methods. Beyond achieving the best average balanced accuracy (BACC) for half of the datasets, the method increased the confidence of the predictions since it presented the lowest loss among all methods. However, the improvement comes with a cost. Training and running multiple deep learning

models demand computational resources. In some cases, this approach is not feasible, since the hardware may not have such resources. Therefore, using an ensemble is basically a trade-off between performance and resources.

Next, we presented a case study on skin cancer detection based on clinical images and patient demographics. In this study, we presented the PAD-UFES-20, a new dataset composed of six common skin lesions – three skin cancers and three skin diseases – composed of images collected from smartphones and patient clinical information. This dataset was collected along with the Dermatological Assistance Program (PAD) at the Federal University of Espírito Santo (UFES) and we made it open in order to support skin cancer research. In this case study, we first applied both combination methods to handle the images and meta-data. In general, both methods provided a significant improvement to performance when compared to the deep learning models without using the meta-data. In addition, the MetaBlock seems to be a more stable method and generally presented a better perfomance than MetaNet and concatenation approaches. Quantitatively, there was an improvement of around 9% on the average balanced accuracy. Lastly, we used the results achieved by the combination approaches to create an ensemble of models and applied the LewDir method to provide online weights for the aggregation approach. In general, the method improved the average balanced accuracy in around 1% for this task. However, it presented similar performance as the remaining aggregation functions used in the experiments.

Despite the promising results achieved by the proposed approaches, there are many points of improvement that may be addressed in the future. We would like to mention a few:

- Both the concatenation and MetaBlock approaches do not identify the importance of each meta-data feature. In addition, they cannot handle the uncertainty on the data. We believe these are important avenues that might be explored.
- The proposed method to determine dynamic weights for an ensemble of deep models uses only the posterior distribution provided by the softmax. It might be an advantage since the method may be used by different types of models. However, we believe we can achieve better performance by using the intermediate feature maps to extract performance metrics. Then, we may use these metrics to identify the miss and hit scores to determine the weights.
- Bias is an important issue in deep learning models that affects skin cancer detection. It is natural to hypothesize that this problem also extends to the meta-data. Thus, we think it is important to investigate this problem in the future.

Another general subject related to skin cancer detection that we believe to be a great

avenue to the future is the models' interpretability. Currently, the most common way that models provide the diagnosis is selecting the label that produces the highest probability. However, how can a clinician interpret a low probability assigned to a melanoma? In fact, they require more explanations than only the model's predictions. In general, a clinician is interested in CAD systems that support their diagnostic by presenting insights and visual explanations of the features used by the model in the classification process. They want to know why the model is selecting such disease. In this sense, we also need to focus on models that are able to output not only the labels' probabilities but the pattern analysis as well. While it is very desirable it is also a challenging task that should be addressed in the future.

Finally, another aspect we believe will become a trend in the near future is the use of three types of skin cancer images: clinical, dermoscopy, and histopathological. Each of these images presents different characteristics, which may help to correlate features to improve the predicted diagnosis. In addition, CAD systems will be able to act from clinical diagnosis to biopsy, which makes it more desirable and useful.

References

Abdi, H.; Williams, L. J. Principal component analysis. *Computational Statistics*, v. 2, n. 4, p. 433–459, 2010.

ACS. *Cancer Facts & Figures 2019*. 2019. American Cancer Society. Last accessed: 10 March 2020. Available on: <https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2019/cancer-facts-and-figures-2019.pdf>.

Akram, T. et al. A deep heterogeneous feature fusion approach for automatic land-use classification. *Information Sciences*, v. 467, p. 199–218, 2018.

Angelo, G. G. de; Pacheco, A. G. C.; Krohling, R. A. Skin lesion segmentation using deep learning for images acquired from smartphones. In: *IEEE International Joint Conference on Neural Networks*. 2019. p. 1–8.

Ardila, D. et al. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature Medicine*, v. 25, n. 6, p. 954, 2019.

Argenziano, G. et al. Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions: comparison of the ABCD rule of dermatoscopy and a new 7-point checklist based on pattern analysis. *Archives of Dermatology*, American Medical Association, v. 134, n. 12, p. 1563–1570, 1998.

Argenziano, G.; Soyer, H. P. Dermoscopy of pigmented skin lesions – a valuable tool for early. *The Lancet Oncology*, v. 2, n. 7, p. 443–449, 2001.

Arroyo, J. L. G.; Zapirain, B. G. Detection of pigment network in dermoscopy images using supervised machine learning and structural analysis. *Computers in Biology and Medicine*, v. 44, p. 144–157, 2014.

Arroyo, R.; Alcantarilla, P. F.; Bergasa, L. M.; Romera, E. Fusion and binarization of CNN features for robust topological localization across seasons. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2016. p. 4656–4663.

Atrey, P. K.; Hossain, M. A.; Saddik, A. E.; Kankanhalli, M. S. Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems*, v. 16, n. 6, p. 345–379, 2010.

Attia, M.; Hossny, M.; Nahavandi, S.; Yazdabadi, A. Skin melanoma segmentation using recurrent and convolutional neural networks. *IEEE 14th International Symposium on Biomedical Imaging*, p. 292–296, 2017.

Audebert, N.; Saux, B. L.; Lefèvrey, S. Fusion of heterogeneous data in convolutional networks for urban semantic labeling. In: *IEEE Joint Urban Remote Sensing Event*. 2017. p. 1–4.

Azulay, R. D. *Dermatologia*. 7. ed. Rio de Janeiro, Brazil: Guanabara Koogan, 2017.

Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Barata, C.; Celebi, M. E.; Marques, J. S. Improving dermoscopy image classification using color constancy. *IEEE Journal of Biomedical and Health Informatics*, v. 19, n. 3, p. 1146–1152, 2014.

Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L. V. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, v. 110, n. 3, p. 346–359, 2008.

Bengio, Y. Practical recommendations for gradient-based training of deep architectures. *Neural networks: Tricks of the trade*, p. 437–478, 2012.

Bengio, Y.; LeCun, Y. et al. Scaling learning algorithms towards AI. *Large-scale Kernel Machines*, v. 34, n. 5, p. 1–41, 2007.

Bengio, Y. et al. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, Now Publishers, Inc., v. 2, n. 1, p. 1–127, 2009.

Berner, C. et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bray, F. et al. Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, v. 68, n. 6, p. 394–424, 2018.

Breiman, L. Bagging predictors. *Machine Learning*, v. 24, n. 2, p. 123–140, 1996.

Brinker, T. J. et al. Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task. *European Journal of Cancer*, v. 113, p. 47–54, 2019.

Brinker, T. J. et al. Skin cancer classification using convolutional neural networks: Systematic review. *Journal of Medical Internet Research*, v. 20, n. 10, p. e11936, 2018.

Britto Jr, A. S.; Sabourin, R.; Oliveira, L. E. Dynamic selection of classifiers - a comprehensive review. *Pattern Recognition*, v. 47, n. 11, p. 3665–3680, 2014.

Brownlee, J. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Melbourne, Australia: Machine Learning Mastery, 2018.

Brun, A. L.; Britto, A. S.; Oliveira, L. S.; Enembreck, F.; Sabourin, R. Contribution of data complexity features on dynamic classifier selection. In: *IEEE International Joint Conference on Neural Networks*. 2016. p. 4396–4403.

Castro, P. B. C.; Krohling, B. A.; Pacheco, A. G.; Krohling, R. A. An app to detect melanoma using deep learning: An approach to handle imbalanced data based on evolutionary algorithms. *IEEE International Joint Conference on Neural Networks*, p. 1–8, 2020.

Cauchy, A. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, v. 25, n. 1847, p. 536–538, 1847.

CCA. *Understanding Skin Cancer - A guide for people with cancer, their families and friends*. 2018. Cancer Council Australia. Last accessed 10 March 2020. Available on: <https://www.cancer.org.au/about-cancer/types-of-cancer/skin-cancer.html>.

CCS. *Canadian Cancer Statistics 2014 - Special topic: Skin cancers*. 2014. Canadian Cancer Society's Advisory Committee on Cancer Statistics. Last accessed: 10 March 2020. Available on: <https://www.cancer.ca/statistics>.

Celebi, M. E. et al. A methodological approach to the classification of dermoscopy images. *Computerized Medical Imaging and Graphics*, v. 31, n. 6, p. 362–373, 2007.

Cevikalp, H.; Polikar, R. Local classifier weighting by quadratic programming. *IEEE Transactions on Neural Networks*, v. 19, n. 10, p. 1832–1838, 2008.

Chai, Y.; Liu, H.; Xu, J. Glaucoma diagnosis based on both hidden features and domain knowledge through deep learning models. *Knowledge-Based Systems*, v. 161, p. 147–156, 2018.

Chaplot, D. S.; Lee, L.; Salakhutdinov, R.; Parikh, D.; Batra, D. Embodied multimodal multitask learning. *arXiv preprint arXiv:1902.01385*, 2019.

Chiu, C.-C. et al. State-of-the-art speech recognition with sequence-to-sequence models. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018. p. 4774–4778.

Chollet, F. *Deep Learning with Python*. Nova York, USA: Manning Publications, 2018. v. 1.

Codella, N. et al. Deep learning, sparse coding, and SVM for melanoma recognition in dermoscopy images. In: Springer. *International Workshop on Machine Learning in Medical Imaging*. 2015. p. 118–126.

Codella, N. et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging, hosted by the International Skin Imaging Collaboration (ISIC). *arXiv:1710.05006*, 2017.

Codella, N. C. et al. Deep learning ensembles for melanoma recognition in dermoscopy images. *IBM Journal of Research and Development*, v. 61, n. 4, p. 5–1, 2017.

Combalia, M. et al. BCN20000: Dermoscopic lesions in the wild. *arXiv:1908.02288*, 2019.

Cruz, R. M.; Sabourin, R.; Cavalcanti, G. D. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, v. 41, p. 195–216, 2018.

Cruz, R. M.; Sabourin, R.; Cavalcanti, G. D.; Ren, T. I. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, v. 48, n. 5, p. 1925–1935, 2015.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, v. 2, n. 4, p. 303–314, 1989.

Dargan, S.; Kumar, M.; Ayyagari, M. R.; Kumar, G. A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, p. 1–22, 2019.

Deng, J. et al. Imagenet: A large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2009. p. 248–255.

Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, v. 1, n. 1, p. 3–18, 2011.

Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dragomir, S. S.; Agarwal, R. P.; Barnett, N. S. Inequalities for beta and gamma functions via some classical and new integral inequalities. *Journal of Inequalities and Applications*, v. 5, n. 2, p. 103–165, 2000.

Duarte, A. F.; Sousa-Pinto, B.; Haneke, E.; Correia, O. Risk factors for development of new skin neoplasms in patients with past history of skin cancer: A survival analysis. *Scientific Reports*, v. 8, n. 1, p. 1–6, 2018.

Ercal, F.; Chawla, A.; Stoecker, W. V.; Lee, H.-C.; Moss, R. H. Neural network diagnosis of malignant melanoma from color images. *IEEE Transactions on Biomedical Engineering*, v. 41, n. 9, p. 837–845, 1994.

Esteva, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, v. 542, n. 7639, p. 115, 2017.

Faes, L. et al. Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study. *The Lancet Digital Health*, v. 1, n. 5, p. e232–e242, 2019.

Fatima, M.; Pasha, M. et al. Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications*, v. 9, n. 01, p. 1, 2017.

Feng, H.; Berk-Krauss, J.; Feng, P. W.; Stein, J. A. Comparison of dermatologist density between urban and rural counties in the United States. *JAMA Dermatology*, American Medical Association, v. 154, n. 11, p. 1265–1271, 2018.

Finlayson, G. D.; Trezzi, E. Shades of gray and colour constancy. In: Society for Imaging Science and Technology. *Color and Imaging Conference*. 2004. p. 37–41.

Frasson, P. H. L. et al. Profile of skin cancer in Pomeranian communities of the state of Espírito Santo. *Revista do Colégio Brasileiro de Cirurgiões*, v. 44, n. 2, p. 187–193, 2017.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, v. 36, n. 4, p. 193–202, 1980.

Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: *International Conference on Machine Learning*. 2016. p. 1050–1059.

Galassi, A.; Lippi, M.; Torroni, P. Attention, please! a critical review of neural attention models in natural language processing. *arXiv preprint arXiv:1902.02181*, 2019.

Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. California, USA: O'Reilly Media, 2019.

Gessert, N.; Nielsen, M.; Shaikh, M.; Werner, R.; Schlaefer, A. Skin lesion classification using ensembles of multi-resolution efficientNets with meta data. *arXiv preprint arXiv:1910.03910*, 2019.

Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*. Cambridge, Massachusetts, USA: MIT press, 2016.

Goodfellow, I. et al. Generative adversarial nets. In: *Advances in Neural Information Processing Systems*. 2014. p. 2672–2680.

Green, A.; Martin, N.; Pfitzner, J.; O'Rourke, M.; Knight, N. Computer image analysis in the diagnosis of melanoma. *Journal of the American Academy of Dermatology*, v. 31, n. 6, p. 958–964, 1994.

Greenacre, M. J. *Theory and Applications of Correspondence Analysis*. London, United Kingdom: London Academic Press, 1984.

Haenssle, H. A. et al. Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Annals of Oncology*, v. 29, n. 8, p. 1836–1842, 2018.

Hai, J. et al. Multi-level features combined end-to-end learning for automated pathological grading of breast cancer on digital mammograms. *Computerized Medical Imaging and Graphics*, v. 71, p. 58–66, 2019.

Han, S. S. et al. Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. *Journal of Investigative Dermatology*, v. 138, n. 7, p. 1529–1538, 2018.

Harangi, B. Skin lesion classification with ensembles of deep convolutional neural networks. *Journal of Biomedical Informatics*, v. 86, p. 25–32, 2018.

Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin, Germany: Springer Science & Business Media, 2009.

Haykin, S. *Neural Networks and Learning Machines*. 3. ed. New Jersey, USA: Pearson Education Inc., 2010.

He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016. p. 770–778.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, v. 14, n. 8, p. 1771–1800, 2002.

Hinton, G. E.; Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, v. 313, n. 5786, p. 504–507, 2006.

Ho, T. K.; Basu, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 3, p. 289–300, 2002.

Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation*, v. 9, n. 8, p. 1735–1780, 1997.

Huang, G. et al. Snapshot ensembles: Train 1, get m for free. In: *International Conference on Learning Representations (ICLR)*. 2017. p. 1–14.

Huang, G.; Liu, Z.; Maaten, L. V. D.; Weinberger, K. Q. Densely connected convolutional networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017. p. 4700–4708.

Hubel, D. H.; Wiesel, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, v. 148, n. 3, p. 574, 1959.

INCA. *The cancer incidence in Brazil*. 2020. National Institute of Cancer José Alencar Gomes (INCA). Last accessed 13 March 2020. Available on: <https://www.inca.gov.br/sites/ufu.sti.inca.local/files//media/document//estimativa-2020-incidencia-de-cancer-no-brasil.pdf>.

Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Irvin, J. et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In: *AAAI Conference on Artificial Intelligence*. 2019. v. 33, p. 590–597.

ISIC. *Skin Lesion Analysis Towards Melanoma Detection*. 2019. International Skin Imaging Collaboration. Last accessed: 10 March 2020. Available on: <https://www.isic-archive.com>.

Ivakhnenko, A. G.; Lapa, V. G. *Cybernetic Predicting Devices*. Purdue University, School of Electrical Engineering. West Lafayette, Indiana, USA, 1966.

James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*. Berlin, Germany: Springer, 2013.

Jeffcock, P. *What's the Difference Between AI, Machine Learning, and Deep Learning?* 2018. Last accessed: 16 March 2020. Available on: <https://blogs.oracle.com/bigdata/difference-ai-machine-learning-deep-learning>.

Jégou, H.; Douze, M.; Schmid, C.; Pérez, P. Aggregating local descriptors into a compact image representation. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2010. p. 3304–3311.

Jing, L.; Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Kabbai, L.; Abdellaoui, M.; Douik, A. Image classification by combining local and global features. *The Visual Computer*, v. 35, n. 5, p. 679–693, 2019.

Kather, J. N. et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLOS Medicine*, v. 16, n. 1, p. e1002730, 2019.

Kermany, D. S. et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, v. 172, n. 5, p. 1122–1131, 2018.

Kharazmi, P.; Kalia, S.; Lui, H.; Wang, Z.; Lee, T. A feature fusion system for basal cell carcinoma detection through data-driven feature learning and patient profile. *Skin Research and Technology*, v. 24, n. 2, p. 256–264, 2018.

Kittler, H.; Pehamberger, H.; Wolff, K.; Binder, M. Diagnostic accuracy of dermoscopy. *The Lancet Oncology*, v. 3, n. 3, p. 159–165, 2002.

Krawczyk, B.; Woźniak, M. Untrained weighted classifier combination with embedded ensemble pruning. *Neurocomputing*, v. 196, p. 14–22, 2016.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*. 2012. p. 1097–1105.

Krogh, A.; Hertz, J. A. A simple weight decay can improve generalization. In: *Advances in Neural Information Processing Systems*. 1992. p. 950–957.

Krohling, R. A.; Pacheco, A. G. A-TOPSIS – an approach based on TOPSIS for ranking evolutionary algorithms. *Procedia Computer Science*, v. 55, p. 308–317, 2015.

Kullback, S.; Leibler, R. A. On information and sufficiency. *The Annals of Mathematical Statistics*, JSTOR, v. 22, n. 1, p. 79–86, 1951.

Kuncheva, L. I. *Combining Pattern Classifiers: Methods and Algorithms*. New Jersey, USA: John Wiley & Sons, 2014.

Larochelle, H.; Hinton, G. E. Learning to combine foveal glimpses with a third-order Boltzmann machine. In: *Advances in neural information processing systems*. 2010. p. 1243–1251.

LeCun, Y.; Bengio, Y. et al. Convolutional networks for images, speech, and time series. *The handbook of Brain Theory and Neural Networks*, n. 10, p. 1–12, 1995.

LeCun, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998.

Li, C.; Wu, X.; Zhao, N.; Cao, X.; Tang, J. Fusing two-stream convolutional neural networks for RGB-T object tracking. *Neurocomputing*, v. 281, p. 78–85, 2018.

Li, F.; Johnson, J.; Yeung, S. et al. *Convolutional Neural Networks for Visual Recognition CS231n*. Stanford University, 2019. Last accessed: 15 March 2020. Available on: <http://cs231n.stanford.edu/>.

Li, K.; Wu, Z.; Peng, K.-C.; Ernst, J.; Fu, Y. Guided attention inference network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 42, p. 2996–3010, 2019.

Li, W.; Zhuang, J.; Wang, R.; Zhang, J.; Zheng, W.-S. Fusing metadata and dermoscopy images for skin disease diagnosis. In: *IEEE International Symposium on Biomedical Imaging*. 2020. p. 1996–2000.

Lin, G.; Fan, G.; Kang, X.; Zhang, E.; Yu, L. Heterogeneous feature structure fusion for classification. *Pattern Recognition*, v. 53, p. 1–11, 2016.

Lin, T.-Y.; RoyChowdhury, A.; Maji, S. Bilinear CNN models for fine-grained visual recognition. In: *IEEE International Conference on Computer Vision*. 2015. p. 1449–1457.

Litjens, G. et al. A survey on deep learning in medical image analysis. *Medical Image Analysis*, v. 42, p. 60–88, 2017.

Liu, Y.; Chen, X.; Cheng, J.; Peng, H. A medical image fusion method based on convolutional neural networks. In: *IEEE International Conference on Information Fusion*. 2017. p. 1–7.

Liu, Y. et al. A deep learning system for differential diagnosis of skin diseases. *arXiv preprint arXiv:1909.05382*, 2019.

Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015. p. 3431–3440.

Lovelace, A. C. Translator's notes to an article on Babbage's analytical engine. *Scientific Memoirs*, v. 3, p. 691–731, 1842.

Lowe, D. G. Object recognition from local scale-invariant features. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 1999. v. 2, p. 1150–1157.

Maaten, L. V. D.; Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, v. 9, p. 2579–2605, 2008.

Maesschalck, R. D.; Jouan-Rimbaud, D.; Massart, D. L. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, v. 50, n. 1, p. 1–18, 2000.

Maglogiannis, I.; Delibasis, K. K. Enhancing classification accuracy utilizing globules and dots features in digital dermoscopy. *Computer Methods and Programs in Biomedicine*, v. 118, n. 2, p. 124–133, 2015.

Majtner, T.; Yildirim-Yayilgan, S.; Hardeberg, J. Y. Combining deep learning and hand-crafted features for skin lesion classification. In: *IEEE Conference on Image Processing Theory, Tools and Applications*. 2016. p. 1–6.

Marghoob, A. A.; Usatine, R. P.; Jaimes, N. *Dermoscopy – two step algorithm*. 2020. Last accessed: 17 June 2020. Available on: <https://usatinemedia.com/app/dermoscopy-two-step-algorithm/>.

Masood, A.; Al-Jumaily, A. A. Computer aided diagnostic support system for skin cancer: a review of techniques and algorithms. *International Journal of Biomedical Imaging*, Hindawi, v. 2013, n. ID 323268, p. 1–22, 2013.

Masoudnia, S.; Ebrahimpour, R. Mixture of experts: a literature survey. *Artificial Intelligence Review*, v. 42, n. 2, p. 275–293, 2014.

Mavrovouniotis, M.; Yang, S. Training neural networks with ant colony optimization algorithms for pattern classification. *Soft Computing*, v. 19, n. 6, p. 1511–1522, 2015.

Mayor, A. *Gods and Robots: Myths, Machines, and Ancient Dreams of Technology*. Princeton, New Jersey: Princeton University Press, 2018.

Merz, C. J. Using correspondence analysis to combine classifiers. *Machine Learning*, v. 36, n. 1-2, p. 33–58, 1999.

Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Miller, G. F.; Todd, P. M.; Hegde, S. U. Designing neural networks using genetic algorithms. In: *International Conference on Genetic Algorithms*. 1989. v. 89, p. 379–384.

Minka, T. *Estimating a Dirichlet distribution*. 2012. MIT Technical report. Last accessed: 15 March 2020. Available on: <https://tminka.github.io/papers/dirichlet/minka-dirichlet.pdf>.

Minsky, M. Steps toward artificial intelligence. *IEEE proceedings of the IRE*, v. 49, n. 1, p. 8–30, 1961.

Minsky, M.; Papert, S. *Perceptrons*. Cambridge, USA: MIT Press, 1969. v. 1.

Mitchell, T. M. *Machine learning*. New York, USA: McGraw-hill, 1997.

Myung, I. J. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, v. 47, n. 1, p. 90–100, 2003.

Nanni, L.; Costa, Y. M.; Lumini, A.; Kim, M. Y.; Baek, S. R. Combining visual and acoustic features for music genre classification. *Expert Systems with Applications*, v. 45, p. 108–117, 2016.

Nassif, A. B.; Shahin, I.; Attili, I.; Azzeh, M.; Shaalan, K. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, v. 7, p. 19143–19165, 2019.

Ng, A. et al. Sparse autoencoder. *CS294A Lecture notes*, Stanford University, v. 72, p. 1–19, 2011.

Ng, K. W.; Tian, G.-L.; Tang, M.-L. *Dirichlet and Related Distributions: Theory, Methods and Applications*. New Jersey, USA: John Wiley & Sons, 2011. v. 888.

Nguyen, D. T.; Pham, T. D.; Baek, N. R.; Park, K. R. Combining deep and handcrafted image features for presentation attack detection in face recognition systems using visible-light camera sensors. *Sensors*, v. 18, n. 3, p. 699, 2018.

Nguyen, T. T.; Liew, A. W.-C.; Tran, M. T.; Pham, X. C.; Nguyen, M. P. A novel genetic algorithm approach for simultaneous feature and classifier selection in multi classifier system. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. 2014. p. 1698–1705.

Nguyen, T. T.; Luong, A. V.; Dang, M. T.; Liew, A. W.-C.; McCall, J. Ensemble selection based on classifier prediction confidence. *Pattern Recognition*, v. 100, p. 104–107, 2020.

Nida, N.; Irtaza, A.; Javed, A.; Yousaf, M. H.; Mahmood, M. T. Melanoma lesion detection and segmentation using deep region based convolutional neural network and fuzzy c-means clustering. *International Journal of Medical Informatics*, v. 124, p. 37–48, 2019.

Niu, X.-X.; Suen, C. Y. A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern Recognition*, v. 45, n. 4, p. 1318–1325, 2012.

Oh, K.; Chung, Y.-C.; Kim, K. W.; Kim, W.-S.; Oh, I.-S. Classification and visualization of Alzheimer's disease using volumetric convolutional neural network and transfer learning. *Scientific Reports*, v. 9, n. 1, p. 1–16, 2019.

Olah, C. *Understanding LSTM Networks*. 2015. Last accessed: 18 March 2020. Available on: <http://colah.github.io/posts/2015-08-Understanding-LSTMs>.

Oliveira, D. V.; Cavalcanti, G. D.; Sabourin, R. Online pruning of base classifiers for dynamic ensemble selection. *Pattern Recognition*, v. 72, p. 44–58, 2017.

Oliveira, R. B.; Papa, J. P.; Pereira, A. S.; Tavares, J. M. R. Computational methods for pigmented skin lesion classification in images: review and future trends. *Neural Computing and Applications*, v. 29, n. 3, p. 613–636, 2018.

Ortega, J. D. et al. Multimodal fusion with deep neural networks for audio-video emotion recognition. *arXiv preprint arXiv:1907.03196*, 2019.

Pacheco, A. G.; Ali, A.-R.; Trappenberg, T. Skin cancer detection based on deep learning and entropy to detect outlier samples. In: *Medical Image Computing and Computer Assisted Intervention (MICCAI) at Skin Lesion Analysis Towards Melanoma Detection (ISIC) challenge*. 2019. p. 1–6.

Pacheco, A. G.; Krohling, R. A. Aggregation of neural classifiers using Choquet integral with respect to a fuzzy measure. *Neurocomputing*, v. 292, p. 151–164, 2018.

Pacheco, A. G.; Krohling, R. A. An approach to improve online sequential extreme learning machines using restricted Boltzmann machines. In: *IEEE International Joint Conference on Neural Networks*. 2018. p. 1–8.

Pacheco, A. G.; Krohling, R. A. Recent advances in deep learning applied to skin cancer detection. In: *Neural Information Processing Systems at Retrospectives workshop*. 2019. p. 1–8.

Pacheco, A. G.; Krohling, R. A. The impact of patient clinical information on automated skin cancer detection. *Computers in Biology and Medicine*, v. 116, p. 103545, 2020.

Pacheco, A. G.; Krohling, R. A. Learning dynamic weights for an ensemble of deep models applied to medical imaging classification. In: *IEEE International Joint Conference on Neural Networks*. 2020. p. 1–8.

Pacheco, A. G.; Krohling, R. A.; Silva, C. A. da. Restricted Boltzmann machine to determine the input weights for extreme learning machines. *Expert Systems with Applications*, v. 96, p. 77–85, 2018.

Pacheco, A. G. et al. PAD-UFES-20: a skin lesion dataset composed of patient data and clinical images collected from smartphones. *Data in Brief*, v. 32, p. 1–10, 2020.

Pacheco, A. G.; Sastry, C. S.; Trappenberg, T.; Oore, S.; Krohling, R. A. On out-of-distribution detection algorithms with deep neural skin cancer classifiers. In: *IEEE Computer Vision and Pattern Recognition (CVPR) at ISIC Skin Image Analysis Workshop*. 2020. p. 1–10.

Pacheco, A. G. C.; Krohling, R. An attention-based mechanism to combine images and metadata in deep learning models applied to skin cancer classification. *IEEE journal of biomedical and health informatics*, 2021. In press.

Pacheco, A. G. C. et al. *PAD-UFES-20: a skin lesion dataset composed of patient data and clinical images collected from smartphones*. 2020. Mendeley Data, v1. Available on: <http://dx.doi.org/10.17632/zr7vgbcyr2.1.>

Partalas, I.; Tsoumakas, G.; Vlahavas, I. P. Focused ensemble selection: A diversity-based method for greedy ensemble selection. In: *European Conference on Artificial Intelligence*. 2008. p. 117–121.

Pedregosa, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

Perez, F.; Avila, S.; Valle, E. Solo or ensemble? choosing a CNN architecture for melanoma classification. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019. p. 1–9.

Perez, F.; Vasconcelos, C.; Avila, S.; Valle, E. Data augmentation for skin lesion analysis. In: *Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*. 2018. p. 303–311.

Perronnin, F.; Sánchez, J.; Mensink, T. Improving the Fisher kernel for large-scale image classification. In: *European Conference on Computer Vision*. 2010. p. 143–156.

Pogorelov, K. et al. Deep learning and handcrafted feature based approaches for automatic detection of angiectasia. In: *IEEE International Conference on Biomedical & Health Informatics*. 2018. p. 365–368.

Polikar, R. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, v. 6, n. 3, p. 21–45, 2006.

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Networks*, v. 12, n. 1, p. 145–151, 1999.

Qummar, S. et al. A deep learning ensemble approach for diabetic retinopathy detection. *IEEE Access*, v. 7, p. 150530–150539, 2019.

Quost, B.; Masson, M.-H.; Denœux, T. Classifier fusion in the Dempster–Shafer framework using optimized t-norm based combination rules. *International Journal of Approximate Reasoning*, v. 52, n. 3, p. 353–374, 2011.

Radford, A. et al. Language models are unsupervised multitask learners. *OpenAI Blog*, v. 1, n. 8, p. 9, 2019.

Raghu, M.; Zhang, C.; Kleinberg, J.; Bengio, S. Transfusion: Understanding transfer learning for medical imaging. In: *Advances in Neural Information Processing Systems*. 2019. p. 3342–3352.

Rawat, W.; Wang, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, v. 29, n. 9, p. 2352–2449, 2017.

Rere, L. R.; Fanany, M. I.; Arymurthy, A. M. Simulated annealing algorithm for deep learning. *Procedia Computer Science*, v. 72, n. 1, p. 137–144, 2015.

Rodrigues, F.; Markou, I.; Pereira, F. C. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion*, v. 49, p. 120–129, 2019.

Rodríguez, P.; Gonfaus, J. M.; Cucurull, G.; XavierRoca, F.; Gonzalez, J. Attend and rectify: a gated attention mechanism for fine-grained recovery. In: *European Conference on Computer Vision (ECCV)*. 2018. p. 349–364.

Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015. p. 234–241.

Ronning, G. Maximum likelihood estimation of Dirichlet distributions. *Journal of Statistical Computation and Simulation*, v. 32, n. 4, p. 215–221, 1989.

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

Rosenblatt, F. *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*. Cornell Aeronautical Lab Inc. Buffalo, NY, USA, 1961.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. *Learning internal representations by error propagation*. University of California, San Diego, California, USA, 1985.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986.

Sabour, S.; Frosst, N.; Hinton, G. E. Dynamic routing between capsules. In: *Advances in Neural Information Processing Systems*. 2017. p. 3856–3866.

Sabourin, M.; Mitiche, A.; Thomas, D.; Nagy, G. Classifier combination for hand-printed digit recognition. In: *IEEE International Conference on Document Analysis and Recognition*. 1993. p. 163–166.

Sahlsten, J. et al. Deep learning fundus image analysis for diabetic retinopathy and macular edema grading. *Scientific reports*, v. 9, n. 1, p. 1–11, 2019.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *IEEE Conference on Computer Vision and Pattern Pecognition*. 2018. p. 4510–4520.

Santos, E. M. D.; Sabourin, R. Classifier ensembles optimization guided by population oracle. In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. 2011. p. 693–698.

Schapire, R. E.; Freund, Y.; Bartlett, P.; Lee, W. S. et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, v. 26, n. 5, p. 1651–1686, 1998.

Scharcanski, J.; Celebi, M. E. *Computer Vision Techniques for the Diagnosis of Skin Cancer*. Berlin, Germany: Springer Science & Business Media, 2013.

Scheffler, R. M.; Liu, J. X.; Kinfu, Y.; Poz, M. R. D. Forecasting the global shortage of physicians: an economic-and needs-based approach. *Bulletin of the World Health Organization*, SCIELO Public Health, v. 86, p. 516–523B, 2008.

Schmidhuber, J. Learning factorial codes by predictability minimization. *Neural Computation*, v. 4, n. 6, p. 863–879, 1992.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks*, v. 61, p. 85–117, 2015.

Sebah, P.; Gourdon, X. Introduction to the gamma function. *American Journal of Scientific Research*, p. 2–18, 2002.

Sejnowski, T. J. *The Deep Learning Revolution: Artificial Intelligence Meets Human Intelligence*. Cambridge, Massachusetts, USA: MIT press, 2018.

Serte, S.; Demirel, H. Gabor wavelet-based deep learning for skin lesion classification. *Computers in Biology and Medicine*, p. 103423, 2019.

Shen, L. et al. Deep learning to improve breast cancer detection on screening mammography. *Scientific reports*, v. 9, n. 1, p. 1–12, 2019.

Siegel, R. L.; Miller, K. D.; Jemal, A. Cancer statistics, 2019. *CA: A Cancer Journal for Clinicians*, v. 69, n. 1, p. 7–34, 2019.

Sierra, S.; González, F. A. Combining textual and visual representations for multimodal author profiling. *Working Notes Papers of the CLEF*, v. 2125, p. 219–228, 2018.

Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sinz, C. et al. Accuracy of dermatoscopy for the diagnosis of nonpigmented cancers of the skin. *Journal of the American Academy of Dermatology*, v. 77, n. 6, p. 1100–1109, 2017.

Smola, A.; Vishwanathan, S. *Introduction to Machine Learning*. Cambridge, United Kingdom: Cambridge University Press, 2008.

Solem, J. E. *Programming Computer Vision with Python: Tools and Algorithms for Analyzing Images*. Cambridge, USA: O'Reilly Media, Inc., 2012.

Spyridonos, P.; Gaitanis, G.; Likas, A.; Bassukas, I. D. Automatic discrimination of actinic keratoses from clinical photographs. *Computers in Biology and Medicine*, v. 88, p. 50–59, 2017.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014.

Staudemeyer, R. C.; Morris, E. R. Understanding LSTM – a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

Štefka, D.; Holeňa, M. Dynamic classifier aggregation using interaction-sensitive fuzzy measures. *Fuzzy Sets and Systems*, v. 270, p. 25–52, 2015.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. A. Inception-v4, Inception-Resnet and the impact of residual connections on learning. In: *AAAI Conference on Artificial Intelligence*. 2017. p. 4278–4284.

Szegedy, C. et al. Going deeper with convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015. p. 1–9.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception architecture for computer vision. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016. p. 2818–2826.

Tan, M.; Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Trappenberg, T. *Fundamentals of Computational Neuroscience*. New York, USA: Oxford University Press, 2009.

Tschandl, P. et al. Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: an open, web-based, international, diagnostic study. *The Lancet Oncology*, v. 20, n. 7, p. 938–947, 2019.

Tschandl, P.; Rosendahl, C.; Kittler, H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Nature Scientific Data*, v. 5, 2018.

Turing, A. Computer machinery and intelligence. *Mind - A Quarterly Review of Psychology and Philosophy*, n. 236, p. 433–460, 1950.

Udrea, A. et al. Accuracy of a smartphone application for triage of skin lesions based on machine learning algorithms. *Journal of the European Academy of Dermatology and Venereology*, v. 34, n. 3, p. 648–655, 2020.

Umbaugh, S. E.; Moss, R. H.; Stoecker, W. V.; Hance, G. A. Automatic color segmentation algorithms with application to skin tumor feature identification. *IEEE Engineering in Medicine and Biology Magazine*, v. 12, n. 3, p. 75–82, 1993.

Valle, E. et al. Data, depth, and design: Learning reliable models for skin lesion analysis. *Neurocomputing*, v. 383, p. 303–313, 2020.

Vasconcelos, C. N.; Vasconcelos, B. N. Experiments using deep learning for dermoscopy image analysis. *Pattern Recognition Letters*, 2017.

Vaswani, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. 2017. p. 5998–6008.

Vinyals, O. et al. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, v. 575, n. 7782, p. 350–354, 2019.

Viswanath, S. E. et al. Dimensionality reduction-based fusion approaches for imaging and non-imaging biomedical data: concepts, workflow, and use-cases. *BMC Medical Imaging*, v. 17, n. 1, p. 2, 2017.

Wang, C.; Elazab, A.; Wu, J.; Hu, Q. Lung nodule classification using deep feature fusion in chest radiography. *Computerized Medical Imaging and Graphics*, v. 57, p. 10–18, 2017.

Werner, B. Biópsia de pele e seu estudo histológico: Por quê? para quê? como? *Anais Brasileiros de Dermatologia*, SCIELO Brasil, v. 84, n. 5, p. 507–513, 2009.

WHO. *How common is the skin cancer?* 2019. World Health Organization (WHO). Last accessed: 10 March 2020. Available on: <http://www.who.int/uv/faq/skincancer/en/index1.html>.

Wighton, P.; Lee, T. K.; Lui, H.; McLean, D. I.; Atkins, M. S. Generalizing common tasks in automated skin lesion diagnosis. *IEEE Transactions on Information Technology in Biomedicine*, v. 15, n. 4, p. 622–629, 2011.

Wolff, K.; Johnson, R. A.; Saavedra, A. P.; Roh, E. K. *Fitzpatrick's Color Atlas and Synopsis of Clinical Dermatology*. 8. ed. New York, USA: McGraw-Hill Education, 2017.

Wolpert, D. H. Stacked generalization. *Neural networks*, v. 5, n. 2, p. 241–259, 1992.

Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Computation*, v. 8, n. 7, p. 1341–1390, 1996.

Woods, K.; Kegelmeyer, W. P.; Bowyer, K. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 4, p. 405–410, 1997.

Wu, H.; Zhang, J.; Huang, K.; Liang, K.; Yu, Y. FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv preprint arXiv:1903.11816*, 2019.

Wu, J. M.-T. et al. Applying an ensemble convolutional neural network with Savitzky–Golay filter to construct a phonocardiogram prediction model. *Applied Soft Computing*, v. 78, p. 29–40, 2019.

Wu, X.; Sahoo, D.; Hoi, S. C. Recent advances in deep learning for object detection. *Neurocomputing*, v. 396, p. 39–64, 2020.

Xiao, Y.; Wu, J.; Lin, Z.; Zhao, X. A deep learning-based multi-model ensemble method for cancer prediction. *Computer Methods and Programs in Biomedicine*, v. 153, p. 1–9, 2018.

Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, v. 13, n. 3, p. 55–75, 2018.

Ypma, T. J. Historical development of the Newton–Raphson method. *SIAM review*, v. 37, n. 4, p. 531–551, 1995.

Yu, L.; Chen, H.; Dou, Q.; Qin, J.; Heng, P.-A. Automated melanoma recognition in dermoscopy images via very deep residual networks. *IEEE Transactions on Medical Imaging*, v. 36, n. 4, p. 994–1004, 2017.

Yu, Z. et al. Melanoma recognition in dermoscopy images via aggregated deep convolutional features. *IEEE Transactions on Biomedical Engineering*, v. 66, n. 4, p. 1006–1016, 2019.

Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, A.; Lipton, Z. C.; Li, M.; Smola, A. J. Dive into deep learning. *D2L AI*, Pennsylvania, USA, v. 1, p. 977, 2020.

Zhang, L.; Xie, Y.; Xidao, L.; Zhang, X. Multi-source heterogeneous data fusion. In: *IEEE International Conference on Artificial Intelligence and Big Data*. 2018. p. 47–51.

Zhou, K.; Greenspan, H.; Dinggang, S. *Deep Learning for Medical Image Analysis*. 1. ed. Cambridge, USA: Academic Press, 2017.

APPENDIX A – Software to collect and analyze clinical skin cancer images and patient data

In this appendix, we describe the software developed to collect the PAD-UFES-20 dataset. As presented in Section 5.1, the PAD takes place in 11 different countryside cities in the Espírito Santo state. Most of these places are rural areas and do not have internet access, which is an important requisite to take into account. In this context, the software infrastructure is composed of three main parts: a local web-server, a remote web-server, and a multi-platform smartphone application. Basically, all data is collected using the smartphone application, which locally connects to the local web-server to store the data. After the data collection is done, as soon as the local web-server gets access to the internet, it synchronizes all data with remote one. In the following, we present a brief description of each system.

A.1 Smartphone-based application

In order to collect the data from the patients, we developed a smartphone-based application for both Android[1] and iPhone[2] platforms. The app was developed using React-Native[3], which is an open-source library, based on Javascript[4], for building user interfaces for both platforms. In Figure 47 is depicted some screenshots of the application using the Android version.

The application is used by clinicians and medical students to collect both skin lesion images and clinical data, such as patient's age, lesion's size, the part of the body where the lesion is located, if the lesion itches, among others. After data collection, the application connects to the local web system to transfer all data from the smartphone device to the online server.

[1] Available on <https://play.google.com/store/apps/details?id=ufes.pad.app>
[2] Available on <https://apps.apple.com/br/app/pad-ufes/id1388412988>
[3] <https://reactnative.dev/>
[4] <https://www.javascript.com/>

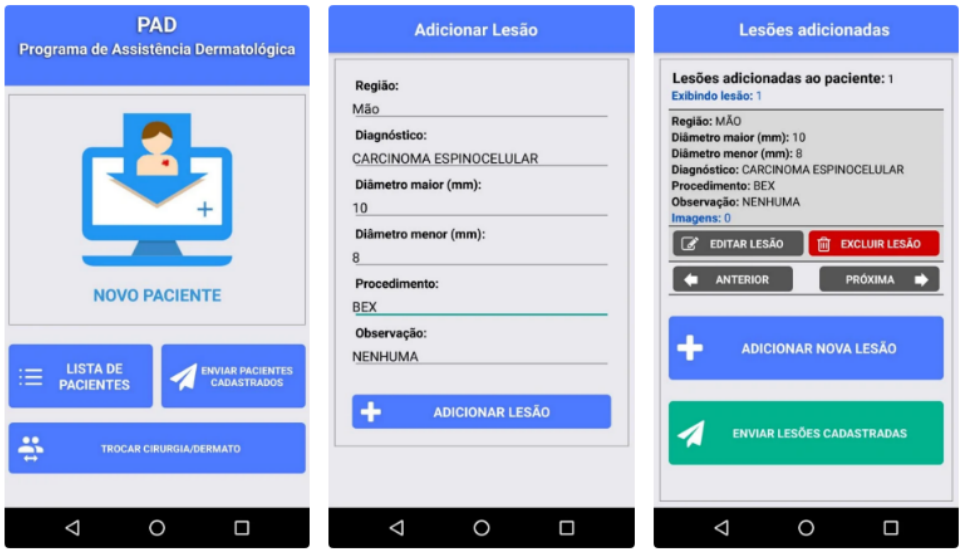


Figure 47 – Screenshots of the smartphone application used to collect skin lesion images and patient data.

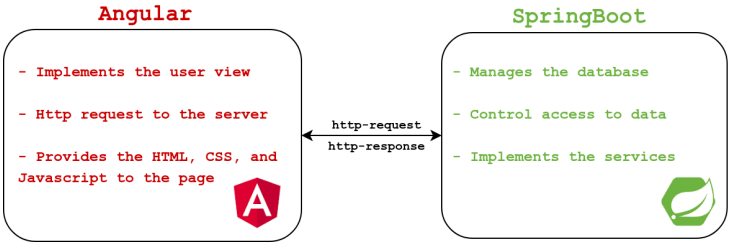


Figure 48 – A schematic diagram describing the relationship between front-end and back-end.

A.2 Web-based system

To store all data collected from PAD, we developed a web-based system[5] that works as a webserver and offers a friendly interface to clinicians to analyze the collected data. The system was developed using two main frameworks: Angular[6] and SpringBoot[7]. Angular is open-source framework, based on Typescript[8], for developing efficient and sophisticated single-page applications. It is used in our front-end side. On the other hand, the back-end was built using SpringBoot[9], which is also an open-source framework, based on Java[10], that is used to create a micro service on the server side. In addition, it is responsible to manage the database system, which is built using MySQL[11]. In Figure 48 is illustrated the relationship between front-end and back-end.

[5] Available on <https://labcin.ufes.br/sade>
[6] <https://angular.io/>
[7] <https://spring.io/projects/spring-boot>
[8] <https://www.typescriptlang.org/>
[9] <https://spring.io/>
[10] <https://www.java.com>
[11] <https://www.mysql.com/>

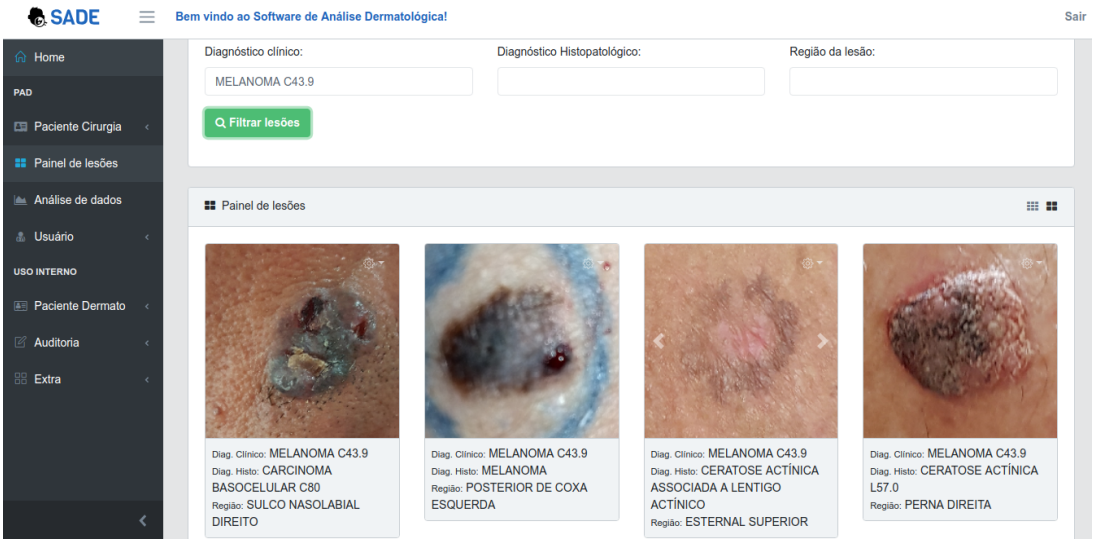


Figure 49 – A screenshot of the web-system used to store and track all skin lesions and patient data.

Beyond store all data in an organized and structured way, this system is important to keep tracking patient lesions, since evolution is an important feature to pay attention on skin cancer detection. Also, it helps clinicians with statistics about lesions and patients. These data are important to understand and improve the quality of the program. Lastly, it is possible to know the dermatologists' performance in detecting skin cancer. Therefore, the system is a significant contribution since it provides important information to PAD and will allow further research in the future.

APPENDIX B – Supplementary material

In this appendix, we present supplementary material regarding the sensitivity analysis of the combination factor – described in Section 3.2 –, ablation tests for the Meta-data Processing Block (MetaBlock) – introduced in Section 3.3 –, and convergence tests to assess the Dirichlet distribution estimation algorithm – described in Section 4.1. Lastly, we present the remaining plots of confusion matrix and ROC curves for the models used in the case study.

B.1 Sensitivity analysis of the combination factor

The goal of this section is to assess the contribution of the extracted features and the meta-data considering the same experimental setup employed to the PAD-UFES-20 case study presented in Chapter 5. As previously described, the total number of clinical features (d_{meta}) after the one-hot-encoding is 81. To vary only the number of selected image features (h) – see Section 3.2 –, we keep d_{meta} and vary c_f from 0.5 to 0.9. In Table 21 is presented the amount of features for each source according to c_f – see Equation 3.4 for more information.

c_f	h	$d_{\mathbf{meta}}$	t_{feat}
0.5	81	81	162
0.6	121	81	202
0.7	189	81	270
0.8	324	81	405
0.9	728	81	809

Table 21 – The number of features from both sources varying the combination factor c_f .

In order to assess each value of c_f presented in Table 21, we trained all five models described in Section 5.2.2, they are EfficientNet-B4, DenseNet-121, MobileNet-v2, ResNet-50, and VGGNet-13, on PAD-UFES-20 dataset considering all values of c_f. The result for each metric is detailed in Table 22. As we can see, the results for all values of c_f presents similar performance, in particular for values above 0.7.

We performed the Friedman test considering the BACC metric for all $c_f \geq 0.7$. The result of the test indicates that there is no statistical difference among these c_f values. In this context, we conclude that any value within $[0.7, 0.9]$ could be used to combine both types of data. We decided to use $c_f = 0.8$ for our experiments because, as previously discussed, the meta-data, in particular for skin cancer detection, should be used as a support source of information. The features extracted from images are still the main source.

$c_f = 0.5$				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-b4	0.772 ± 0.018	0.749 ± 0.038	0.947 ± 0.006	0.655 ± 0.041
DenseNet-121	0.723 ± 0.050	0.73 ± 0.0340	0.925 ± 0.007	0.796 ± 0.076
MobileNet-v2	0.741 ± 0.0170	0.743 ± 0.0200	0.943 ± 0.006	0.646 ± 0.047
ResNet-50	0.719 ± 0.039	0.708 ± 0.038	0.918 ± 0.011	0.845 ± 0.064
VGGNet-13	0.725 ± 0.064	0.728 ± 0.032	0.928 ± 0.008	0.778 ± 0.118
$c_f = 0.6$				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-b4	0.762 ± 0.019	0.759 ± 0.017	0.942 ± 0.004	0.645 ± 0.018
DenseNet-121	0.707 ± 0.043	0.739 ± 0.026	0.926 ± 0.006	0.800 ± 0.094
MobileNet-v2	0.743 ± 0.028	0.76 ± 0.0060	0.938 ± 0.005	0.689 ± 0.064
ResNet-50	0.75 ± 0.0100	0.726 ± 0.014	0.928 ± 0.006	0.827 ± 0.070
VGGNet-13	0.709 ± 0.039	0.735 ± 0.020	0.923 ± 0.003	0.799 ± 0.059
$c_f = 0.7$				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-b4	0.758 ± 0.040	0.761 ± 0.015	0.943 ± 0.005	0.632 ± 0.066
DenseNet-121	0.678 ± 0.028	0.723 ± 0.028	0.919 ± 0.008	0.878 ± 0.110
MobileNet-v2	0.733 ± 0.031	0.751 ± 0.021	0.936 ± 0.004	0.704 ± 0.046
ResNet-50	0.748 ± 0.035	0.731 ± 0.020	0.929 ± 0.007	0.857 ± 0.136
VGGNet-13	0.747 ± 0.024	0.75 ± 0.0120	0.926 ± 0.006	0.744 ± 0.049
$c_f = 0.8$				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-b4	0.765 ± 0.032	0.758 ± 0.012	0.945 ± 0.005	0.637 ± 0.053
DenseNet-121	0.742 ± 0.035	0.747 ± 0.019	0.932 ± 0.005	0.786 ± 0.057
MobileNet-v2	0.738 ± 0.026	0.741 ± 0.017	0.927 ± 0.007	0.767 ± 0.087
ResNet-50	0.741 ± 0.014	0.728 ± 0.029	0.929 ± 0.006	0.833 ± 0.080
VGGNet-13	0.712 ± 0.035	0.720 ± 0.0180	0.929 ± 0.002	0.765 ± 0.051
$c_f = 0.9$				
Model	**ACC**	**BACC**	**AUC**	**Loss**
EfficientNet-b4	0.735 ± 0.038	0.734 ± 0.068	0.938 ± 0.016	0.711 ± 0.017
DenseNet-121	0.723 ± 0.040	0.728 ± 0.028	0.921 ± 0.007	0.868 ± 0.087
MobileNet-v2	0.734 ± 0.023	0.741 ± 0.020	0.932 ± 0.004	0.71 ± 0.0370
ResNet-50	0.747 ± 0.014	0.727 ± 0.018	0.93 ± 0.0050	0.843 ± 0.045
VGGNet-13	0.720 ± 0.024	0.734 ± 0.012	0.919 ± 0.008	0.803 ± 0.034

Table 22 – All CNN models performance considering different values for c_f

.

In this sense, the method assumes that this statement may be achieved by considering 80% of the final features coming from the image.

B.2 Assessing the contribution of each connection gate in the Meta-data Processing Block (MetaBlock)

In this section, we aim to assess the contribution of each gate in the Meta-data Processing Block (MetaBlock) architecture described in Section 3.3. To better understand

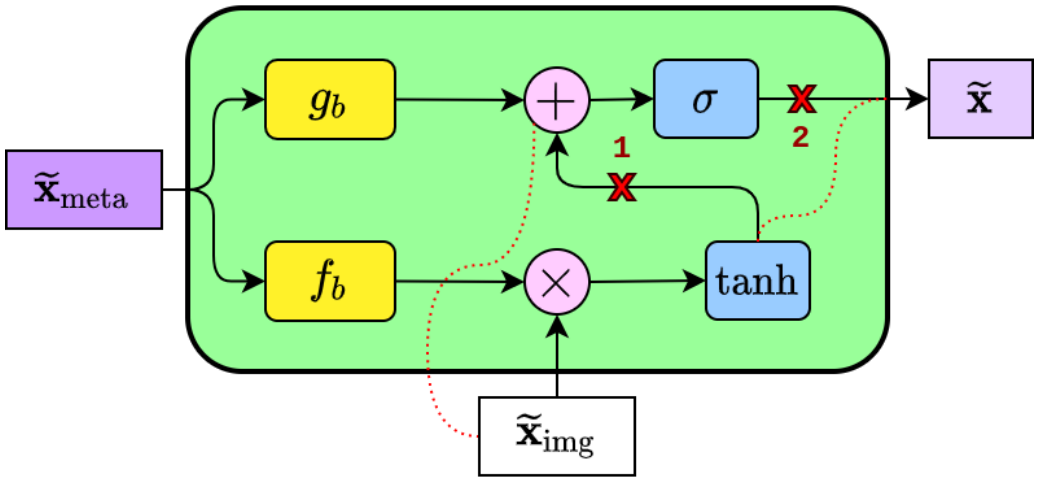


Figure 50 – Adding connection breakers (red X) into the MetaBlock in order to assess the contribution of each gate.

this section, let us modify Figure 21 to get Figure 50. In the previous experiments, we performed the MetaBlock considering both sigmoid and hyperbolic tangent gates. In this section, we perform the block removing a gate connection per time, i.e., first, we perform the block cutting the hyperbolic tangent connection; next, we cut the sigmoid one and connect the tangent directly to the output. These two experiments are illustrated in Figure 50. Imagine each red X as a connection breaker. To remove the sigmoid connection, we activate 1 and 2 and connect tanh to the output. On the other hand, to remove the tangent connection, we activate 1 and deactivate 2. It is worth noting that by removing a gate we are also intentionally removing the scaling and shifting operation, as we can note in the figure.

We perform both experiments using the same Deep Learning models described in the case study. All models are also trained on PAD-UFES-20 dataset considering the modification on the MetaBlock. In Table 23 and 24 are presented the performance of each model considering only the sigmoid connection and the hyperbolic tangent connection, respectively. In Table 25 is presented the comparison of the MetaBlock considering only one and both connections. As we can note, the results considering only the sigmoid connection are slightly better than considering only the hyperbolic tangent one. In general, the average BACC is higher and the loss is lower. Nonetheless, the difference is not enough to conclude that this connection is more relevant to the block. Essentially, both connections are playing important roles in the block. The results in Table 25 support this statement. As we may observe, both connections contribute to MetaBlock for achieving better performance. For all models, except MobileNet-v2, the average BACC is higher when both connections are employed.

Considering only the sigmoid connection				
Model	ACC	BACC	AUC	Loss
EfficientNet-B4	0.655 ± 0.028	0.706 ± 0.021	0.922 ± 0.007	0.856 ± 0.045
DenseNet-121	0.695 ± 0.026	0.728 ± 0.022	0.931 ± 0.004	0.774 ± 0.050
MobileNet-v2	0.759 ± 0.019	0.768 ± 0.024	0.943 ± 0.003	0.642 ± 0.018
ResNet-50	0.696 ± 0.048	0.721 ± 0.024	0.929 ± 0.005	0.797 ± 0.110
VGGNet-13	0.698 ± 0.036	0.733 ± 0.034	0.935 ± 0.009	0.768 ± 0.081

Table 23 – Deep learning models' performance using the MetaBlock considering only the sigmoid connection.

Considering only the hyperbolic tangent connection				
Model	ACC	BACC	AUC	Loss
EfficientNet-B4	0.676 ± 0.023	0.685 ± 0.024	0.897 ± 0.010	0.903 ± 0.076
DenseNet-121	0.695 ± 0.024	0.736 ± 0.013	0.923 ± 0.009	0.828 ± 0.069
MobileNet-v2	0.730 ± 0.026	0.763 ± 0.020	0.937 ± 0.003	0.723 ± 0.031
ResNet-50	0.714 ± 0.024	0.719 ± 0.019	0.934 ± 0.007	0.734 ± 0.056
VGGNet-13	0.686 ± 0.032	0.727 ± 0.002	0.911 ± 0.014	0.869 ± 0.094

Table 24 – Deep learning models' performance using the MetaBlock considering only the tangent hyperbolyc connection.

Model	**Sigmoid**	**Hyperbolic tangent**	**Both**
EfficientNet-B4	0.706 ± 0.021	0.685 ± 0.024	0.770 ± 0.016
DenseNet-121	0.728 ± 0.022	0.736 ± 0.013	0.746 ± 0.039
MobileNet-V2	0.768 ± 0.024	0.763 ± 0.020	0.754 ± 0.014
ResNet-50	0.721 ± 0.024	0.719 ± 0.019	0.765 ± 0.017
VGGNet-13	0.733 ± 0.034	0.727 ± 0.028	0.736 ± 0.018

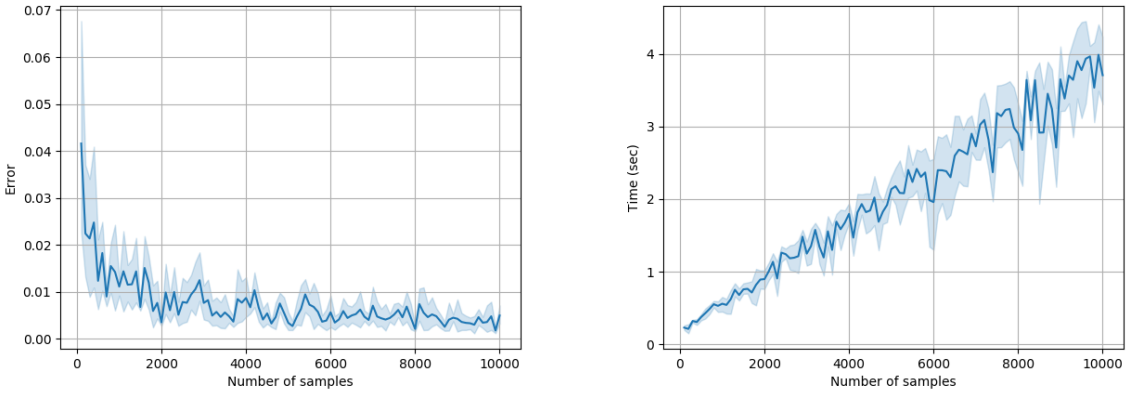
Table 25 – Comparison of the MetaBlock performance in terms of BACC using only the sigmoid, hyperbolic tangent, and both gates.

B.3 The impact of the MetaBlock in the size of the CNN models

In Table 26, we show the impact of the MetaBlock parameters in each CNN model size using the PAD-UFES-20, the dataset that has the higher number of patient demographics available. As we can see, the most impacted model is the VGG-13, in which the MetaBlock increases it by only 0.85%, which is not relevant in terms of the model's training time.

Model	**Original**	**Metablock**	**Impact (%)**
EfficientNet-B4	19,352,374	9,408	0,05%
DenseNet-121	6,960,006	5,376	0,08%
MobileNet-v2	2,231,558	6,720	0,30%
ResNet-50	23,520,326	10,752	0,05%
VGGNet-13	9,611,592	81,534	0,85%

Table 26 – The impact of the MetaBlock parameters in each CNN models size.



(a) Estimation error × number of samples (b) Estimation time × number of samples

Figure 51 – The performance of the Dirichlet distribution estimation algorithm in terms of estimation error and time per number of samples.

B.4 Convergence tests of the Dirichlet distribution estimation algorithm

In this section, we perform experiments to assess the convergence of the algorithm described in Section 4.1.2 to estimate the Dirichlet distribution. We evaluate the algorithm in terms of computational time and estimation error.

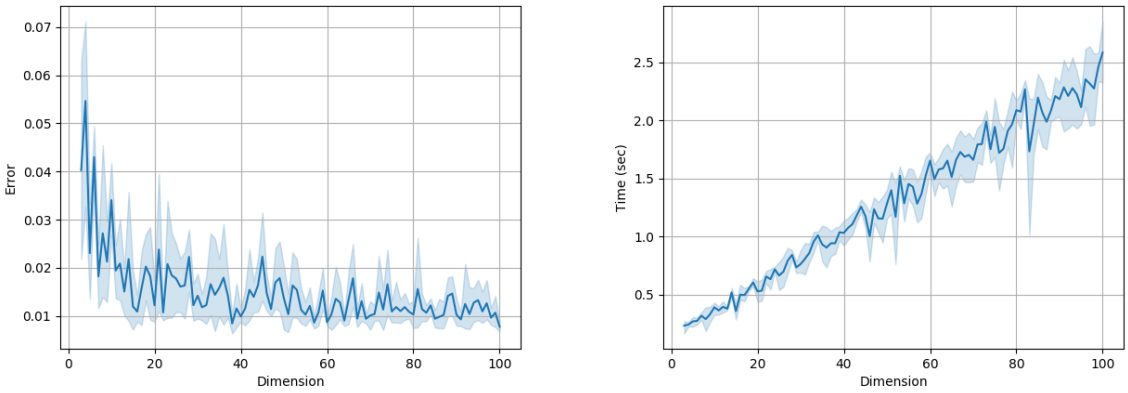
In the first experiments, we randomly select a Dirichlet distribution parameters $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_k\}$, where k is the distribution's dimension. Next, we sample from this distribution to generate a set of data $D = \{\mathbf{d}^1, \cdots, \mathbf{d}^n\}$, where $\mathbf{d} = \{d_1, \cdots, d_k\}$ and n is the number of samples. After sampling, we apply Algorithm 2 to estimate $\hat{\boldsymbol{\alpha}}$ for D and we measure the estimation error using the following relative error:

$$\text{Error} = \frac{|\,\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}\,|}{|\,\hat{\boldsymbol{\alpha}}\,|} \tag{B.1}$$

where $|\cdot|$ represents the vector norm.

We start by setting $k = 5$ and changing the number of samples n from 100 to 10,000 using a step of 100. We repeat this experiment 30 times. In Figure 51 is depicted the plot of the estimator's performance in terms of relative error and computational time per number of samples. We can observe that even for a small number of samples, the algorithm can estimate the distribution with an error of around 0.04. When the number of samples is close to 2,000, the error decreases to around 0.01. In terms of the computational time, for the maximum number of samples, 10,000, the computational time is around 4 seconds[1], which shows the algorithm is pretty fast.

[1] Experiments carried out using a machine with a i7 8Th generation processor and 8GB of RAM



(a) Estimation error × dimension (b) Estimation time × dimension

Figure 52 – The performance of the Dirichlet distribution estimation algorithm in terms of estimation error and time per dimension.

Now, we keep the number of samples $n = 2,000$ and vary the dimension k from 3 to 100 using a step of 1. We also repeat this experiment 30 times. In Figure 52 is shown the plot of the estimator's performance in terms of relative error and computational time per dimension. From the figure, we observe that the relative error starts around 0.05 and decrease for higher dimensions. In terms of time, likewise the previous experiment, for the maximum dimension, the algorithm converges after around 2.5 seconds.

To conclude this section, we perform an experiment to measure the estimation competence for the ISIC 2019 dataset. We perform the ResNet-50 to 4,300 samples in the dataset and use the posterior distribution obtained by the softmax to estimate the Dirichlet distribution for this set. In this case, the dataset is our D with $n = 4,300$ examples and $k = 8$, which is the number of labels within the dataset. After estimating $\hat{\boldsymbol{\alpha}}$, we sample a new dataset $\hat{D}$ from the estimated distribution and compute the Mean Squared Error (MSE) between D and $\hat{D}$ to check if the estimated data is close to the original one. In Figure 53 is shown the scatter plot of both D and $\hat{D}$ using Principal Component Analysis (PCA) to reduce the data dimension from 8 to 2. As we can note, the data inferred from the estimated distribution is close to the original one and overlap the original samples in many cases. This difference is reflected by the MSE error which returned 0.087 for this experiment. The estimation time is around 3 seconds, which is in line with the previous experiments and shows that the algorithm is also fast for non-sintetic data.

B.5 Confusion matrices and ROC curves

In Section 5.2.2 of the case study, when we analyze the results for the combination of image and meta-data approaches, we presented the confusion matrix and ROC curve plots only for ResNet-50. In this section, we present the rest of the plots for DenseNet-121,

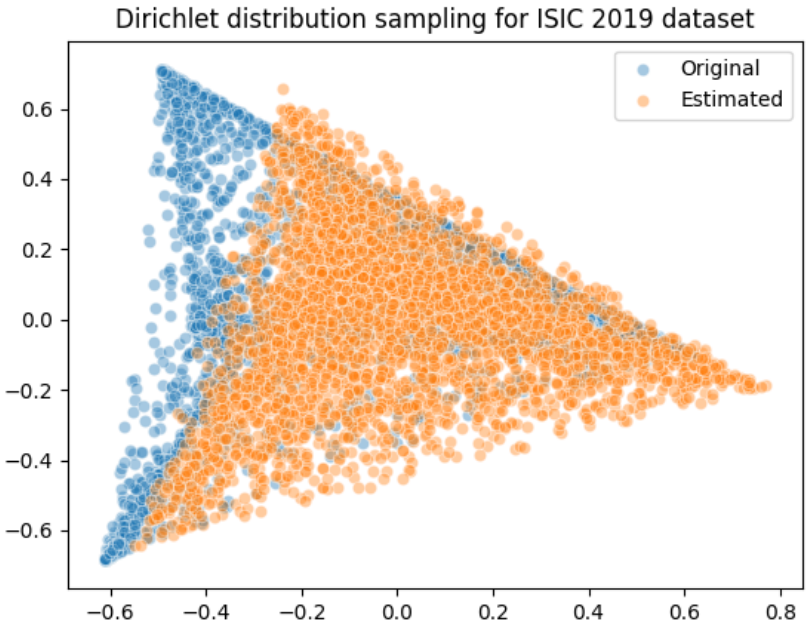
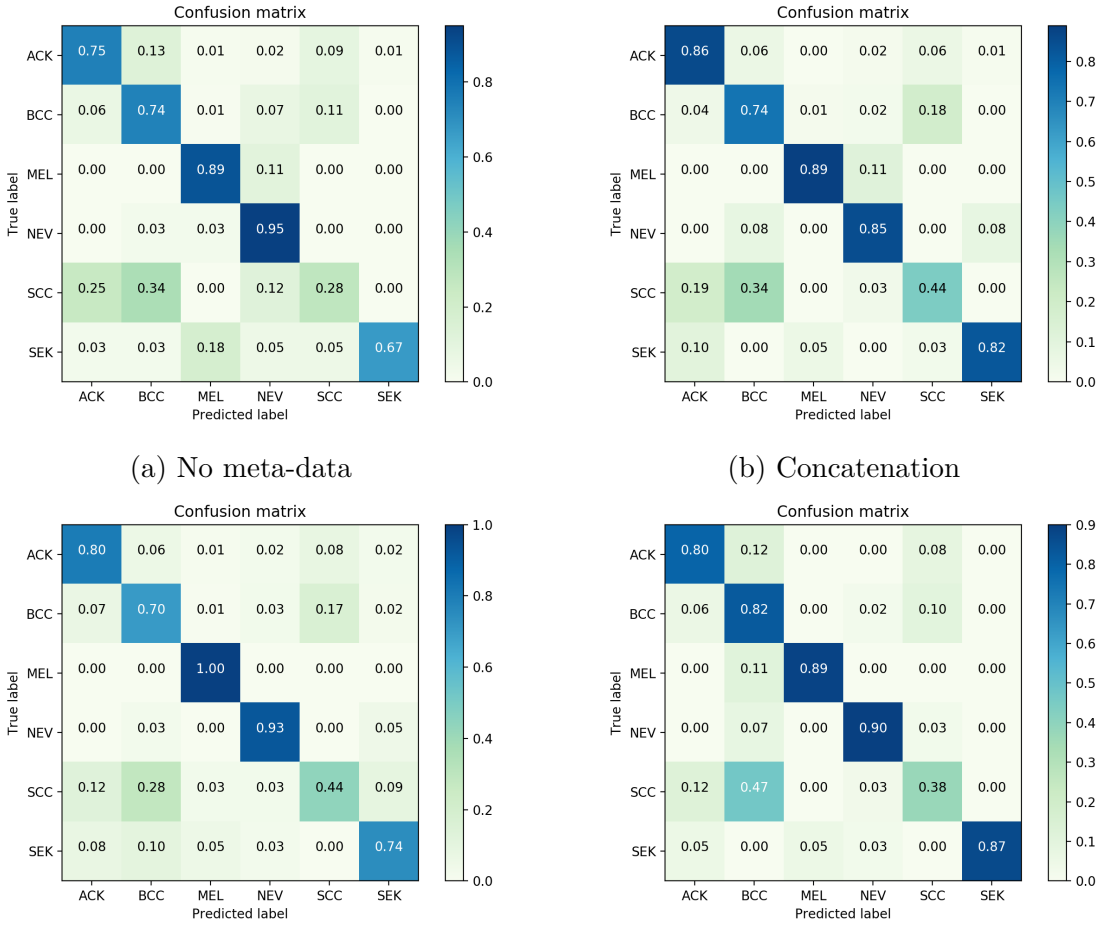


Figure 53 – The scatter plot of the original and estimated data using the Dirichlet distribution.

EfficientNet-b4, MobileNet-v2, and VGGNet-13. The results for these models are similar to the analysis presented for ResNet-50. In general, from Figures 54 to 61, we observe that the combination methods improved the performance mainly to the pigmented skin lesions. Likewise our analysis for ResNet-50, the remaining models are still confusing BCC and SCC quite often. However, recall that this is not a significant problem since both diagnostics are skin cancers and must be referred to biopsy.



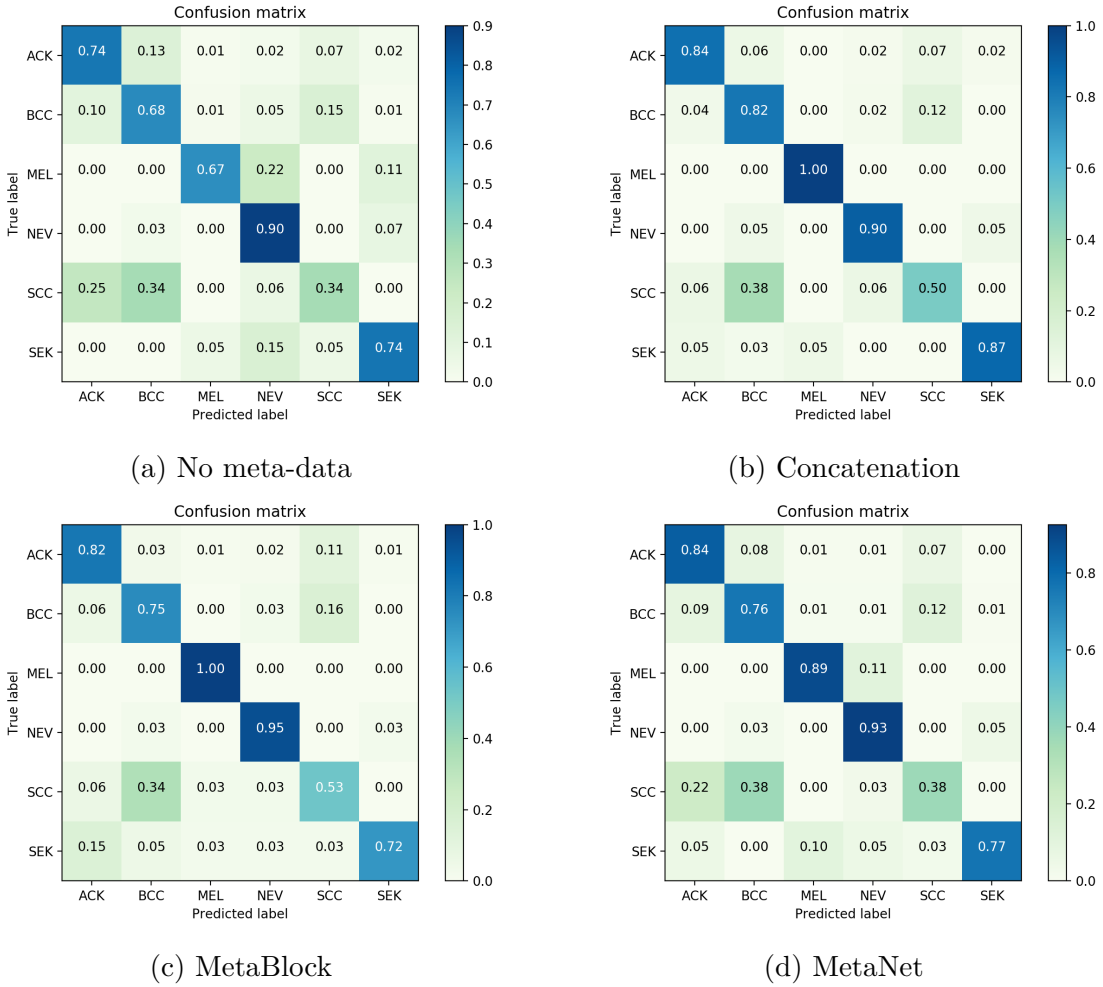
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 54 – Confusion matrices for DenseNet-121 considering all methods.



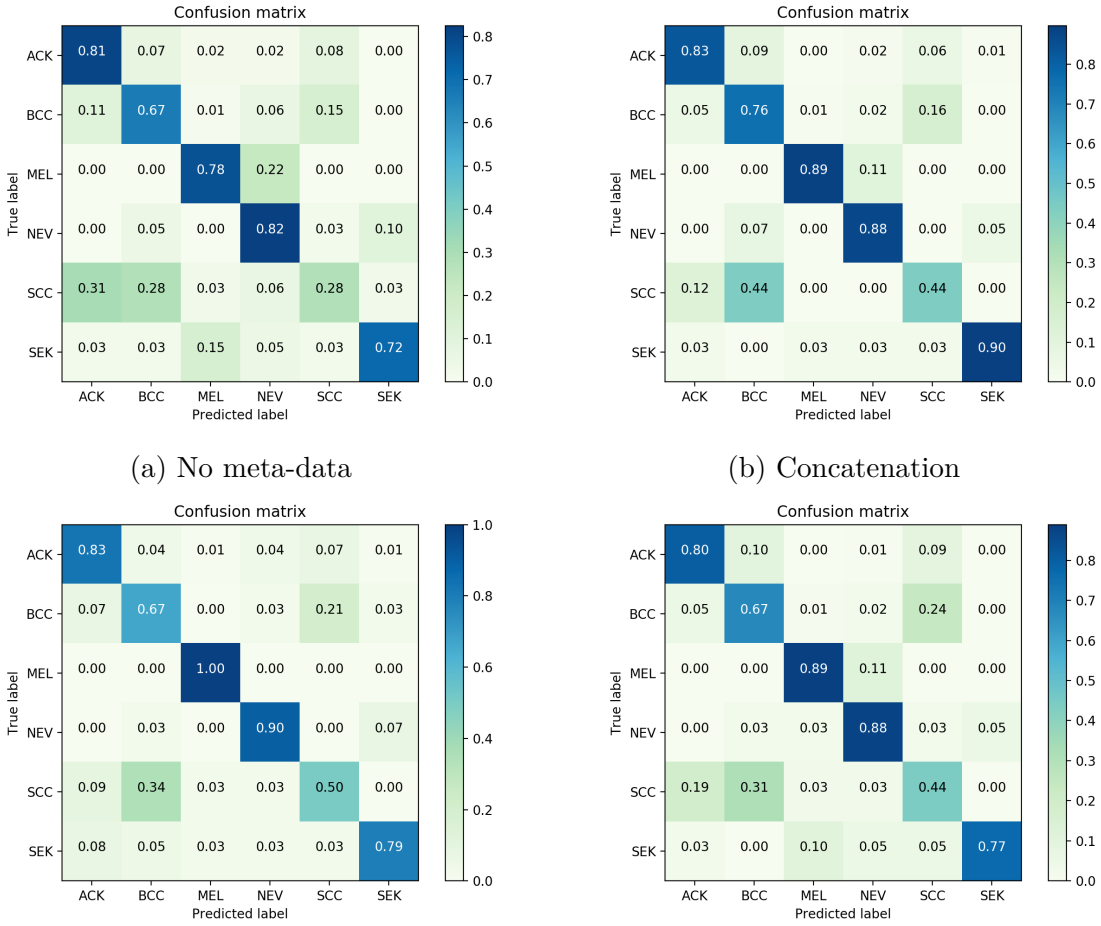
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 55 – Confusion matrices for EfficientNet-b4 considering all methods.



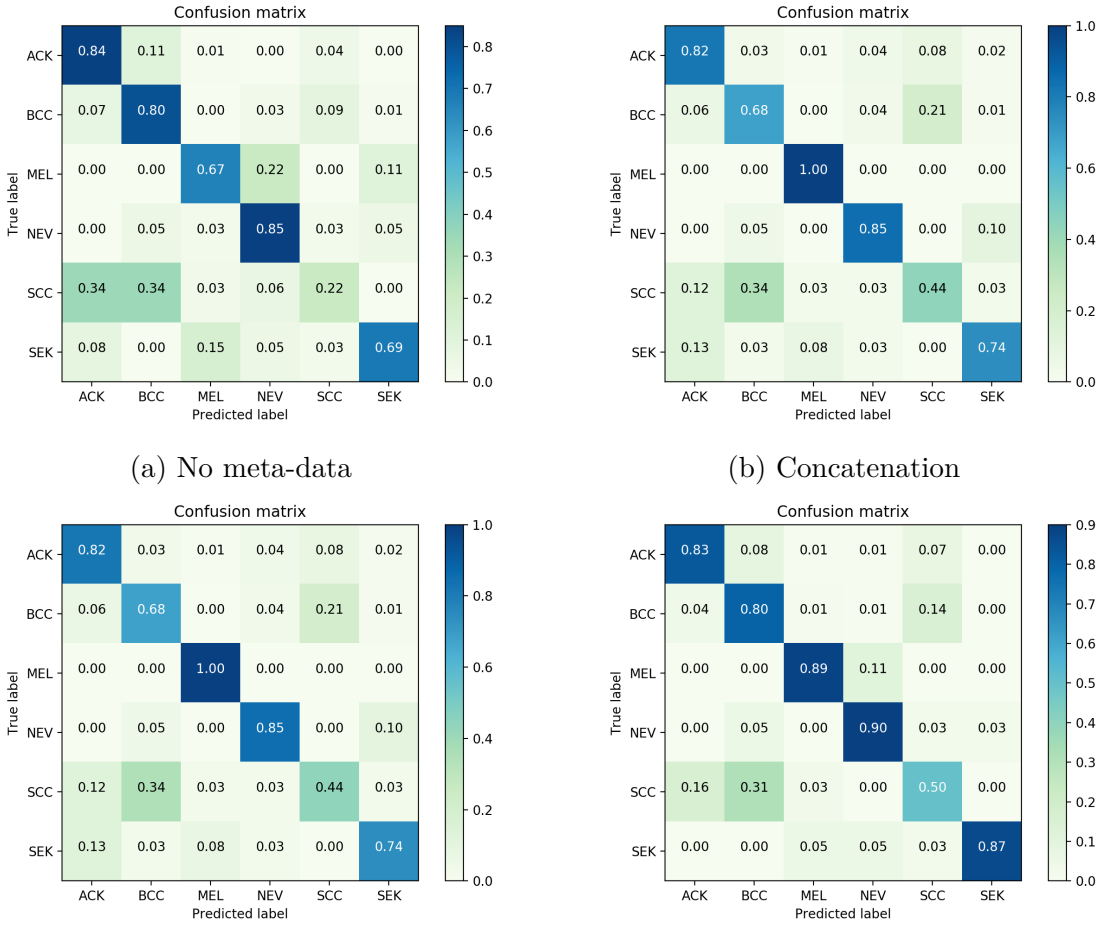
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 56 – Confusion matrices for MobileNet-v2 considering all methods.



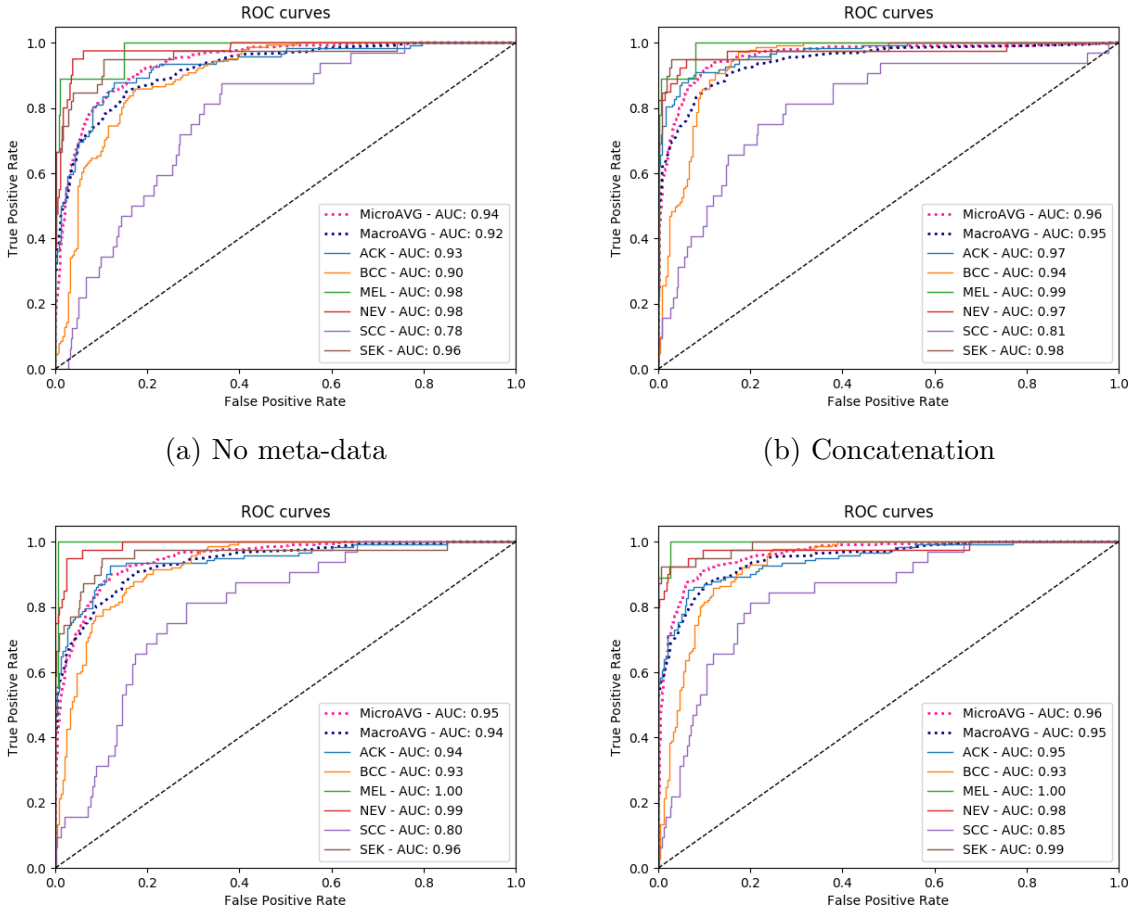
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 57 – Confusion matrices for VGGNet-13 considering all methods.



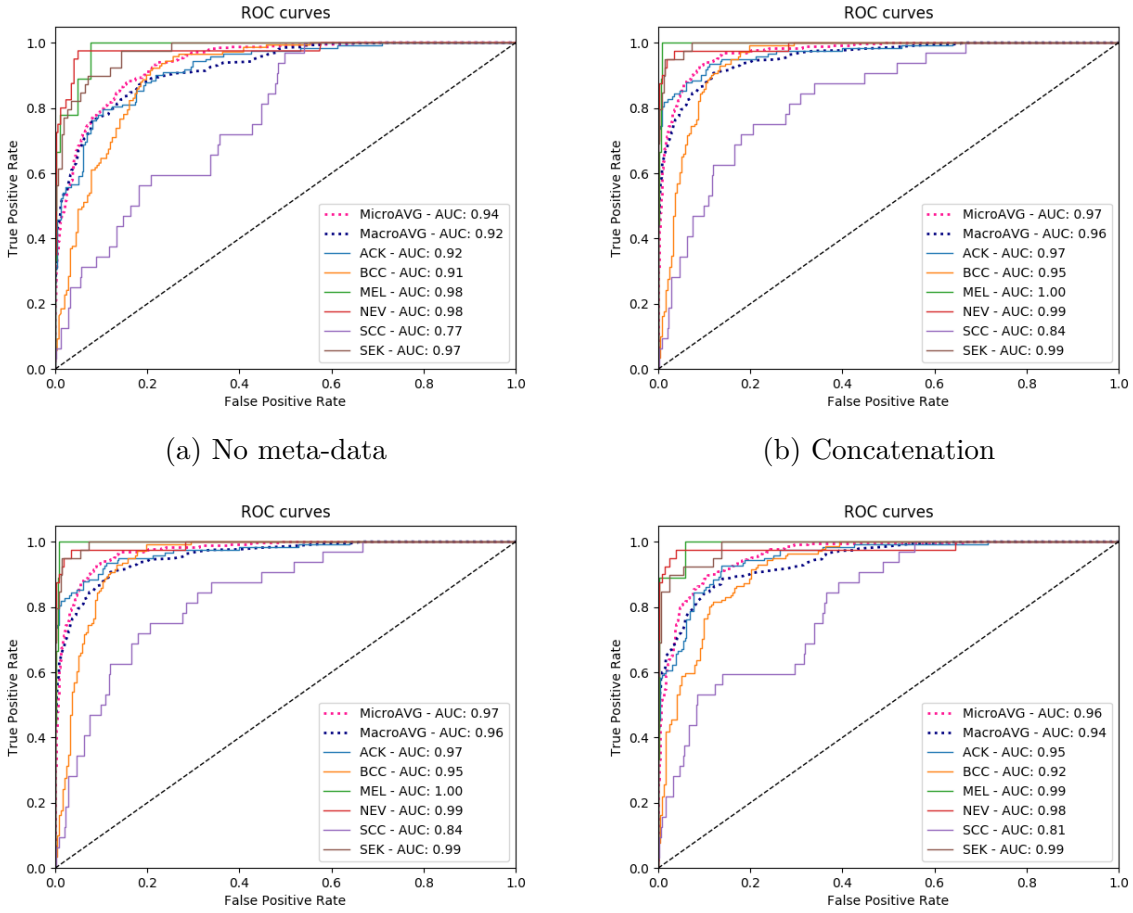
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 58 – ROC curve plots for DenseNet-121 considering all methods.



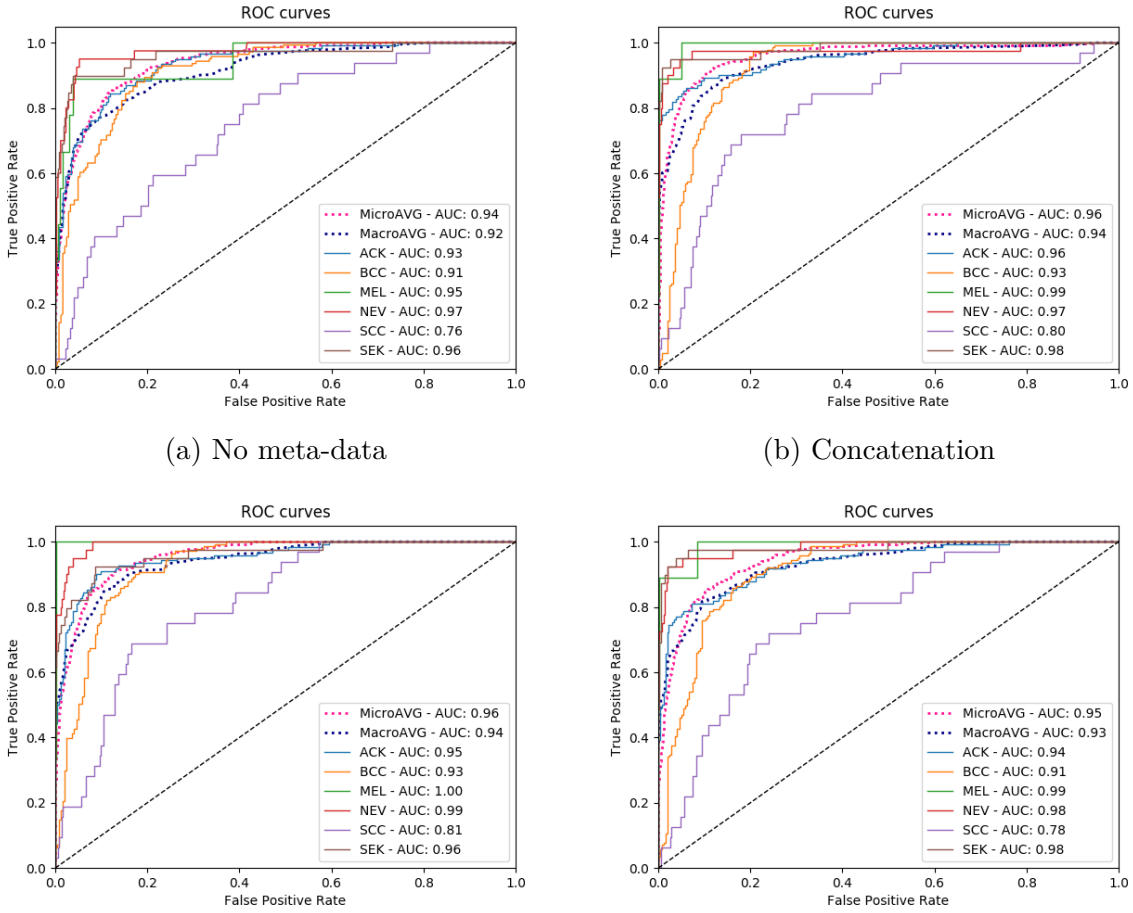
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 59 – ROC curve plots for EfficientNet-b4 considering all methods.



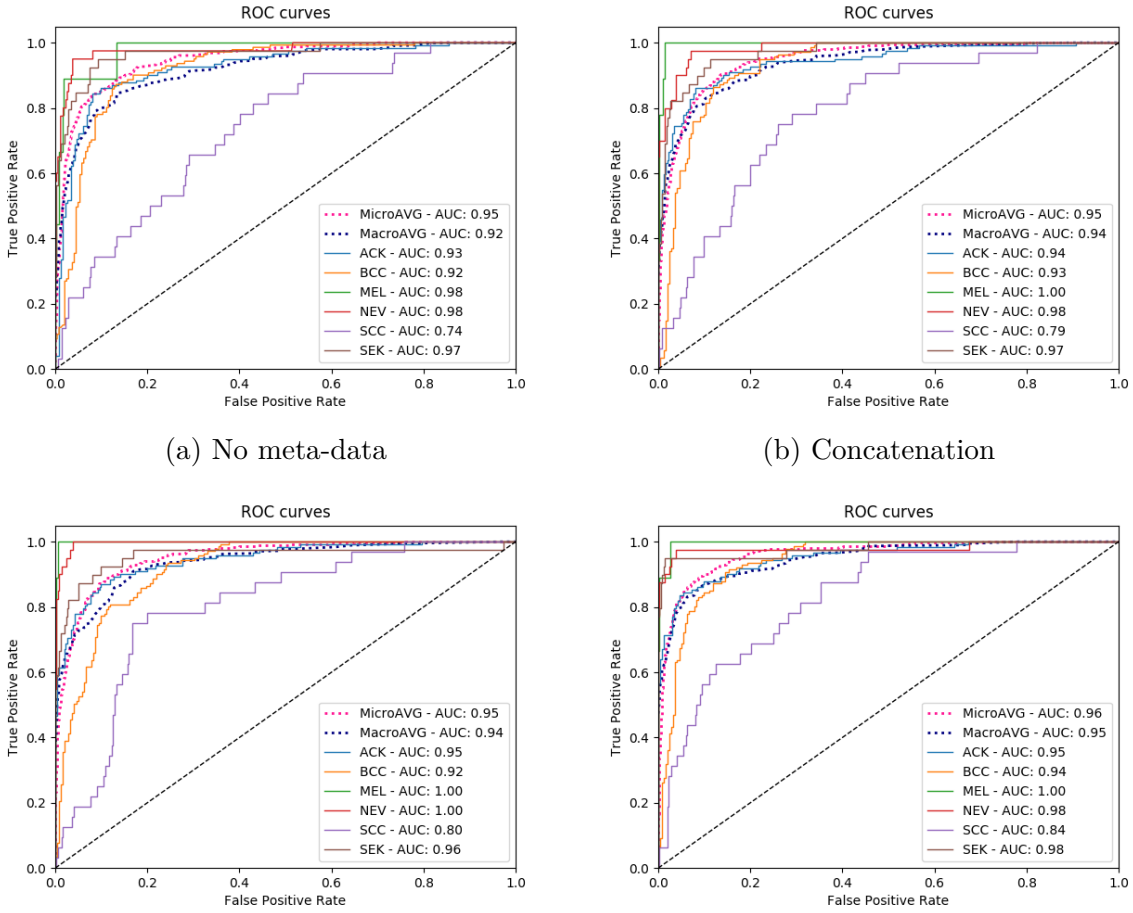
(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 60 – ROC curve plots for MobileNet-v2 considering all methods.



(a) No meta-data

(b) Concatenation

(c) MetaBlock

(d) MetaNet

Figure 61 – ROC curve plots for VGGNet-13 considering all methods.