UCLouvain

epl

École polytechnique de Louvain

Development of an open source application for melanoma image detection

Author: **Loïc Hernaut**
Supervisor: **Sébastien Jodogne**
Readers: **Benoît Duhoux, Jean Vanderdonckt**
Academic year 2025-2026
Master [120] in Computer Science

Abstract

Skin melanoma is one of the most aggressive forms of skin cancer, with early detection being critical to patient prognosis. Despite the growing availability of AI-based diagnostic tools, the majority of high-performance dermatological systems remain proprietary, lacking accessible open-source alternatives that integrate both a specialized deep learning model and a ready-to-use interface.

This thesis presents the full development of an open-source melanoma detection system, from model design to clinical web application. The proposed deep learning model follows a multimodal feature-level fusion architecture, combining a visual feature extractor based on EfficientNet-B6 with a Multi-Layer Perceptron that encodes clinical metadata. The model is trained on the ISIC 2020 dataset and addresses the inherent class imbalance through the use of Focal Loss. Data augmentation and Stratified K-Fold cross-validation are employed to improve generalization and robustness. The trained model achieves a mean AUC of 0.910 across cross-validation folds, with a recall of 0.807, meeting the clinical imperative to minimize false negatives.

Regarding the deployment of the model, the PyTorch network is exported to ONNX format with preprocessing logic encapsulated directly within the computation graph. A mobile-oriented web application enables fully client-side inference via the ONNX Runtime Web library and WebAssembly, guaranteeing that no patient data is transmitted over the network. Experimental evaluation confirms that browser-based inference remains sufficiently responsive for real-world use. The resulting system is therefore practically viable for end-users on their mobile devices. This work delivers a complete, open-source and ready-to-use melanoma detection system.

Acknowledgments

First of all, I would like to express sincere gratitude to my promoter, Sébastien Jodogne, for his guidance throughout the realization of this thesis, and particularly for his availability and supervision throughout the year.

I would also like to express my sincere gratitude to everyone who reviewed part or the entirety of this thesis. I am deeply grateful to Benoît Duhoux, Théodore Cousin, Philippe Hernaut my father, and Mathieu Reniers for dedicating their valuable time to enhancing the quality of this work and for providing constructive feedback.

Finally, I wish to extend my thanks to my family, friends, and entire entourage. Their unconditional support and confidence in me have been my greatest source of strength throughout this academic journey.

Declaration on the Use of AI

Before delving into the core of this thesis, I would like to outline the role of generative AI in this work.

On one hand, I used generative AI for the code, specifically, for the frontend design, to generate graphs based on my data, and to improve code quality. On the other hand, I used generative AI during the writing process to rephrase and translate certain parts of my text.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 1
- 1.2 Proposed Contribution . . . 3
  - 1.2.1 Development of a Deep Learning Model . . . 3
  - 1.2.2 Translation and Deployment of the Model . . . 4

2 Related Work **6**
- 2.1 Datasets . . . 6
  - 2.1.1 ISIC Archive . . . 7
  - 2.1.2 HAM10000 . . . 7
  - 2.1.3 PH^2 Dataset . . . 8
- 2.2 Melanoma Detection Before Deep Learning . . . 8
- 2.3 Deep Learning in Melanoma . . . 9
  - 2.3.1 The Rise of CNNs in Melanoma Diagnosis . . . 9
  - 2.3.2 Mitigating Class Imbalance in Deep Learning . . . 10
  - 2.3.3 Enhancing Model Robustness . . . 11
  - 2.3.4 Multimodal Architectures . . . 12
- 2.4 Deployment of deep learning models . . . 13
  - 2.4.1 Deployment architectures . . . 13

3 Development of a deep learning model for melanoma **15**
- 3.1 Data Preparation . . . 15
  - 3.1.1 Preprocessing . . . 15
  - 3.1.2 Data Augmentation . . . 16
- 3.2 Model Architecture . . . 17
  - 3.2.1 EfficientNet Architecture . . . 18
  - 3.2.2 Visual Feature Extraction . . . 20
  - 3.2.3 Clinical Metadata Encoding . . . 20
  - 3.2.4 Multimodal Fusion and Classification . . . 22
- 3.3 Training Strategy . . . 22
  - 3.3.1 Handling Class Imbalance . . . 23
  - 3.3.2 Optimization . . . 24

4 Model Deployment and Web Integration **25**
4.1 Export with ONNX . . . 26
4.1.1 Introduction to ONNX . . . 26
4.1.2 Export Process . . . 27
4.1.3 Encapsulation of preprocessing . . . 27
4.2 Web Application . . . 28
4.2.1 Technologies and Application Logic . . . 28
4.2.2 Clinical Interface and User Workflow . . . 29
4.3 Inference via WebAssembly . . . 30
4.3.1 Introduction to WebAssembly . . . 30
4.3.2 Running the Model on the Web Application . . . 31

5 Results **32**
5.1 Evaluation metrics . . . 32
5.1.1 AUC . . . 32
5.1.2 Accuracy . . . 33
5.1.3 Recall . . . 33
5.1.4 Precision . . . 33
5.1.5 F1-Score . . . 34
5.2 Model Analysis . . . 34
5.2.1 Experimental Setup . . . 34
5.2.2 Analysis of Model Learning . . . 34
5.2.3 Performance Metrics . . . 36
5.2.4 State-of-the-Art Comparison . . . 36
5.2.5 Results and Discussion . . . 37
5.3 Web Application Analysis . . . 39
5.3.1 Experimental Conditions . . . 39
5.3.2 Model Parity . . . 40
5.3.3 Loading of the Model . . . 41
5.3.4 Inference Latency Analysis . . . 41
5.3.5 Results and Discussion . . . 43

6 Conclusion **44**
6.1 Key Contributions . . . 44
6.2 Results and Discussion . . . 45
6.3 Future Work . . . 46

A Resources and Demo **53**
A.1 Data Availability . . . 53
A.2 Video Demonstration . . . 53

Chapter 1

Introduction

This introductory chapter establishes the foundations of this thesis. It begins by stating the problem, first by contextualizing the medical challenges posed by the classification of skin melanoma, before identifying the specific gap this work addresses. It then outlines the concrete contributions of this work and concludes by outlining the overall structure of the document.

1.1 Problem Statement

Skin melanoma is one of the most aggressive and deadly form of skin cancer. It originates in melanocytes, the cells responsible for producing melanin and the pigment that gives the skin its color. This form of cancer is extremely dangerous due to its ability to spread rapidly to other organs. According to the International Agency for Research on Cancer, in 2022 about 60,000 people died from melanoma[1].

The clinical challenge in dermatology lies in differentiating benign from malignant lesions. Benign lesions, such as nevi (commonly known as moles) are stable and non-invasive cellular proliferations. In contrast, melanoma is characterized by asymmetrical and uncontrolled cell growth that invades the epidermis and subsequently the dermis. The complexity of diagnosis lies in the fact that an early-stage melanoma can be visually indistinguishable from an atypical but benign nevus, as illustrated in Figure 1.1, where the first two images are benign lesions and the last two malignant. Identifying the subtle visual signs of malignancy among a large number of harmless lesions requires specialized clinical expertise.

[1]International Agency for Research on Cancer (IARC). Skin Cancer, `https://www.iarc.who.int/cancer-type/skin-cancer/`

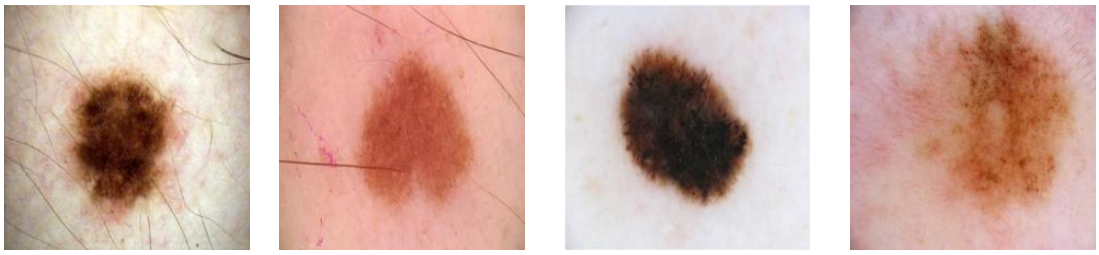


Figure 1.1: Images of skin lesions from ISIC 2020[2]

Early detection is therefore a clinical imperative, as the prognosis is directly correlated with tumor thickness at the time of excision. Historically the visual assessment of lesions relies on cognitive algorithms the most widely used is the ABCDE rule (Asymmetry, Borders, Multiple Colour, Diameter, Evolution) [1]. However the assessment of these criteria remains highly subjective and it is very difficult even for experienced dermatologists to ensure total diagnostic accuracy. In this context, and to address the limitations of human visual assessment the development of diagnostic support systems is particularly relevant. These tools provide practical assistance to practitioners optimizing clinical care and reducing the risk of false negatives.

In response to this need of clinical assistance, artificial intelligence has emerged as a leading technological solution in recent years. In particular, the advent of deep learning has profoundly revolutionized the field of medical image analysis. At the heart of this revolution, Convolutional Neural Networks (CNNs) have demonstrated remarkable ability to process and analyze complex visual data [2]. Thanks to their architecture, these mathematical models are able to extract subtle features from dermoscopic images, identifying patterns of malignancy that are sometimes imperceptible with the naked eye.

However, approaches based only on dermoscopic image analysis have limitations. In clinical practice, dermatologists never rely solely on a visual examination of the lesion. They systematically take into account contextual information specific to the patient, such as age, gender, and the anatomical location of the lesion. Based on this observation, part of the scientific community has turned to multimodal architectures, capable of simultaneously processing heterogeneous data such as images and clinical metadata. This is the direction this work chooses to explore.

[2]ISIC: International Skin Imaging Collaboration, `https://www.isic-archive.com`

Furthermore, putting these models into production is a challenge in itself. While training a high-performing model is a fundamental step, the ultimate goal lies in its deployment and in making it accessible to end users in real-world conditions. In this context, two main deployment approaches are in contrast: Server-Side deployment (Cloud Computing) which offloads computation to remote infrastructure equipped with substantial GPU resources, and Client-Side deployment (Edge Computing), which performs inference directly on the user's device without transmitting data to a third-party server. This second approach offers a significant advantage in the medical field, where patient data confidentiality is a major constraint.

In addition to these architectural considerations, the current state of the art suffers from a scarcity of publicly available solutions. Currently, the majority of high-performance dermatological diagnostic tools remain proprietary, commercialized and locked into closed software ecosystems, as exemplified by applications such as SkinVision[3]. There is a lack of open-source projects that offer a complete end-to-end architecture integrating both a specialized deep learning model for melanoma classification and a ready-to-use web interface. It is precisely this observation that serves as the central motivation for this thesis. The objective of this work is to design a comprehensive system dedicated to melanoma detection and to make it publicly available. This involves not only developing and training the neural network but also integrating it into a functional application, thereby offering a concrete technical alternative to current proprietary solutions.

1.2 Proposed Contribution

To answer to this lack of available solutions, this thesis proposes the full development of a melanoma detection system, from the creation of the machine learning model to the development of a web application for mobile devices.

1.2.1 Development of a Deep Learning Model

First of all, let's focus on the creation of the neural network. The chosen approach relies on a multimodal network capable of simultaneously processing a dermoscopic image together with the patient's clinical metadata. The data used to train the model comes from the ISIC 2020 dataset which contains over 33,000 dermoscopic images [3]. Before being fed into the network, both types of data are standardized and normalized into a format suitable for the neural network. On the image side, this consists of resizing each image to a fixed resolution. On the metadata side, missing values are imputed, the age variable is normalized, and categorical variables

[3]SkinVision Application: `https://www.skinvision.com/`

are one-hot encoded, yielding a compact 11-dimensional input vector. Additionally, a stochastic data augmentation strategy is applied during training to artificially increase dataset diversity and reduce the risk of overfitting. Returning to the architecture, the heart of the model is structured around three major components:

- **Extraction of Visual Features:** The architecture leverages EfficientNet-B6 as its backbone [4]. Rather than training this network from scratch the model is initialized with weights pre-trained on ImageNet [5]. This transfer learning strategy allows the network to inherit robust visual representations and to focus its adaptation on the dermatology-specific patterns present in the training data.
- **The Encoding of Clinical Metadata:** For the metadata, a dedicated Multi-Layer Perceptron (MLP) is designed to encode three clinical variables provided alongside each image in the dataset: the age of the patient, their biological sex, and the anatomical location of the lesion. These variables are then preprocessed before being fed into the sub-network which projects them into a compact latent representation of 32 features.
- **Fusion and Classification:** The representations produced by these two independent streams are then concatenated into a unified vector and passed through a classification head that outputs a single probability of malignancy. This feature-level fusion strategy allows the model to jointly learn the correlations between the visual morphology of the lesion and the patient's clinical context.

Finally, one of the major challenges associated with melanoma classification is the significant class imbalance. In public datasets, malignant cases are underrepresented. To address this and prevent overfitting, several techniques are implemented in this thesis notably through the use of a weighted loss function. This strategy aligns with the clinical imperative to minimize the number of false negatives.

1.2.2 Translation and Deployment of the Model

The second contribution of this work concerns the software engineering required to transform this trained model into an accessible clinical tool. This deployment process is structured around three key stages:

- **Export (ONNX):** The model trained using PyTorch is converted to ONNX (Open Neural Network Exchange) format [6]. This step allows the model to be separated from its training environment and generates a standardized file that is smaller in size and can be easily transferred to another environment.

- **Development of the Web Application:** The user interface is developed entirely in TypeScript. The aim is to create a web application designed for mobile devices that incorporates all the features required for a comprehensive melanoma classification tool. The front-end code thus encapsulates all the application logic: image capture, data preprocessing and integration with the inference engine.
- **Local inference (WebAssembly):** Prediction calculations are performed directly in the user's browser using WebAssembly [7] and the ONNX Runtime Web library [8]. This approach runs the model locally, without requiring a remote server to perform the calculations.

The main advantage of this architecture is the protection of medical data. As the inference is performed entirely on the client side, no images or clinical information are transmitted over the network, thus offering a practical and secure alternative to traditional cloud-based deployments.

Chapter 2

Related Work

This chapter provides the theoretical foundations required to contextualize our work within the existing scientific literature. The objective of this review is to examine the historical and contemporary landscape of automated skin lesion classification.

The main dermoscopic datasets used by the scientific community are examined, along with their shared limitations. Next, classical image analysis methods and handcrafted feature extraction used before the rise of deep learning are presented. The development of CNNs for melanoma detection is then reviewed focusing on class imbalance, ensemble learning, and multimodal architectures. Afterwards, different deployment strategies are explored, contrasting server-side frameworks with client-side execution in the browser.

2.1 Datasets

The training and evaluation of deep learning models rely on dermoscopic image databases annotated by domain experts. These datasets are essential both for training neural networks and for objectively comparing the performance of different architectures. In the field of dermoscopic image analysis, the number of publicly available datasets is limited and those that exist share several common limitations.

First and foremost, the primary challenge to address is a significant class imbalance. Reflecting clinical reality, benign lesions are massively overrepresented compared to malignant cases, which introduces a bias into the training of neural networks. A second issue concerns the size of the available datasets. Model training requires a large number of images in order to achieve high performance. The production of reliable reference labels requires expert dermatologist consensus and

often histopathological[1] confirmation, making large-scale annotation efforts both costly and time-consuming. Additionally, strict patient privacy regulations such as GDPR further restrict data sharing across institutions. As a result, most public datasets remain relatively modest in size, and their limited demographic diversity (particularly the underrepresentation of darker skin tones) raises concerns about the generalization of models trained. Below are the main dermoscopic image databases that are widely used by the scientific community.

2.1.1 ISIC Archive

The ISIC (International Skin Imaging Collaboration) archive[2] is the world's largest open-access collection of dermoscopic images. Contributed to by several international hospitals and research institutions, this archive is regularly expanded through the annual 'ISIC Challenges' which have become a standard benchmark for evaluating and comparing melanoma detection algorithms. It contains tens of thousands of images covering a wide variety of melanocytic and non-melanocytic conditions and remains the primary resource for training large-scale models. Crucially, many editions of the dataset include structured patient-level metadata such as age, sex, and anatomical localization of the lesion. The ISIC 2020 challenge dataset, for instance, contains over 33,000 dermoscopic images from more than 2,000 patients, with all malignant diagnoses confirmed via histopathology [3].

2.1.2 HAM10000

As part of the ISIC archive (ISIC 2018 challenge), the HAM10000 (Human Against Machine with 10000 training images)[3] database is one of the most widely used for academic research. It contains 10 015 dermoscopic images collected over 20 years. The strength of this dataset lies in its reliability and diversity. Firstly, in terms of diagnostic confirmation methods (histopathology, long-term follow-up or expert consensus for benign cases). Secondly, it includes a wide range of represented populations and anatomical areas [9].

[1] Refers to the microscopic examination of a tissue biopsy to confirm a disease diagnosis.
[2] Dataset ISIC: `https://www.isic-archive.com`
[3] Dataset HAM10000: `https://api.isic-archive.com/collections/66/`

2.1.3 PH² Dataset

In contrast to the two datasets presented earlier, the PH² dataset[4] is smaller (200 images) but of very high quality. It originates from the dermatology department at Pedro Hispano Hospital in Portugal. Each image is accompanied by detailed clinical annotations including manual lesion segmentation, dermoscopic feature analysis, and a three-class diagnostic label distinguishing common nevi, atypical nevi, and melanomas. Given its small size, this dataset is rarely used as a standalone training set. Instead, it is frequently employed as an independent test set to evaluate and validate the generalization ability of algorithms trained on larger datasets [10].

2.2 Melanoma Detection Before Deep Learning

The automated analysis of skin lesions has undergone a fundamental technological shift, moving from systems that rely on human expertise to autonomous machine learning models. Historically, before the widespread adoption of deep learning models, the classification of dermoscopic images relied heavily on traditional computer vision methods. This classical approach systematically involved four distinct sequential stages which are detailed below [11].

1. **Image Preprocessing:** Raw images had to be meticulously cleaned to prevent artifacts from being misinterpreted as clinical features. For instance, artifacts like hair were typically filtered out using specialized algorithms such as DullRazor, which applied morphological operations to erase the hair and replace it with a skin texture [12].

2. **Lesion Segmentation:** Following preprocessing, the lesion had to be separated from the healthy surrounding skin using segmentation techniques. To this end, Zhou et al. proposed one of the most effective techniques in this field, employing a Spatially Smoothed Example-Based Classifier (SEBC) algorithm [13].

3. **Handcrafted Feature Extraction:** The core of this methodology rested on handcrafted feature extraction, where researchers had to manually design mathematical algorithms to translate clinical criteria (such as the ABCDE rule) into measurable numerical quantities. For example, the solution proposed by Celebi et al. [14] offers feature extraction techniques that group features into three distinct categories: shape, color, and texture features.

[4]Dataset PH²: www.fc.up.pt/addi/ph2 database

4. **Statistical Classification:** These manually extracted features were subsequently fed into statistical classifiers such as Support Vector Machines (SVMs) or Random Forests to generate a final probability of malignancy.

Despite encouraging results, this methodology had a fundamental structural limitation because it was entirely dependent on researchers' ability to model complex biological patterns. Melanomas display such visual heterogeneity that these handcrafted descriptors often missed the most subtle signs of malignancy. Furthermore, this approach is highly sensitive to errors, as a single error at any step will have irreversible consequences for the rest of the process. For example, poor segmentation, a change in lighting or inadequate hair removal can completely compromise the final results.

The rise of Convolutional Neural Networks (CNNs) revolutionized this paradigm. Unlike classical methods, a CNN requires no explicit segmentation or pre-computed descriptors: it takes raw image pixels as input and, through successive convolutional layers, progressively learns to extract the most discriminative patterns on its own. This ability to bypass the limitations of human-designed models and identify relevant visual features directly from raw data has made deep learning the absolute standard in dermatological imaging today.

2.3 Deep Learning in Melanoma

2.3.1 The Rise of CNNs in Melanoma Diagnosis

The limitations of handcrafted feature extraction were definitively overcome with the advent of Deep Learning. Over the past decade, Convolutional Neural Networks (CNNs) have not only replaced traditional computer vision methods but achieved diagnostic accuracies that rival those of professionals in the sector. In 2017, Esteva et al. marked a turning point in automated skin cancer detection by training an Exception-v3 CNN architecture on a massive and proprietary dataset of 129,450 clinical images [15]. They demonstrated that a single neural network could perform at a dermatologist level across multiple diagnostic tasks including the classification between malignant melanomas and benign nevi.

However, the development of this type of model faces a major obstacle: the lack of high-quality annotated data. Datasets containing hundreds of thousands of images are exceptionally rare in dermatology. To overcome this problem, modern approaches rely heavily on transfer learning. Instead of training models from scratch (random weight initialization), researchers leverage architectures pre-trained on large-scale datasets most commonly used is ImageNet[5]. As a result, the first layers

of these pre-trained models already possess the ability to detect basic concepts. During the fine-tuning phase on dermoscopic datasets (like the ISIC archives) the network only needs to adapt its deeper layers to recognize pathology-specific combinations of these concepts. Following this breakthrough, deep convolutional architectures such as VGG [16], ResNet [17] and DenseNet [18] became the standard baselines for this task. Comparative studies extensively demonstrated their high efficacy in extracting discriminative features from dermatological images [19]. More recently, the focus has shifted toward highly optimized architectures like EfficientNet [4]. Proposed by Google, EfficientNet utilizes a 'compound scaling' method that uniformly scales network width, depth, and resolution. In the context of medical imaging, EfficientNet variants have demonstrated superior feature extraction capabilities compared to other models of this type such as ResNet or MobileNets.

2.3.2 Mitigating Class Imbalance in Deep Learning

Although deep learning model demonstrate high capacity in the field of melanoma classification, their performance on public dermatological datasets is frequently bottlenecked by severe class imbalance with a low proportion of malignant cases. If left unaddressed, a CNN will mathematically bias its predictions toward the majority class to minimize its global error rate. Spelmen and Porkodi detail several existing strategies to mitigate this issue, classified into three distinct categories of techniques: data-level, algorithm-level, and hybrid approaches [20].

At the data level, resampling techniques such as Over-Sampling (OS) and Under-Sampling (US) are commonly employed to rebalance the training distribution before the learning phase. Sayed et al. demonstrate how these data-centric strategies help adjust class proportions [21]. Under-sampling reduces the dominance of the majority class by discarding benign samples, whereas over-sampling expands the minority class by duplicating or generating malignant instances, ensuring the network receives sufficient exposure to rare lesions.

At the algorithm level, class imbalance is managed by modifying the training dynamics through specialized objective functions without altering the underlying data. Le et al. highlight the effectiveness of integrating a focal loss function for automatic skin cancer classification [22]. Rather than treating all misclassifications the same way, focal loss automatically reduces the importance of majority samples, pushing the model to focus its learning on the underrepresented malignant cases.

Hybrid approaches seek to combine the strengths of both data-level and algorithm-level methods, simultaneously adjusting the dataset distribution and modifying the

learning algorithm. Chang et al. implement this strategy by combining K-Means SMOTE with an XGBoost classifier for advanced melanoma detection [23]. Instead of performing simple data replication, their framework operates in two complementary steps. First, at the data level, K-Means SMOTE generates synthetic samples to balance the datasets. Second, at the algorithmic level, an XGBoost classifier trains on this balanced data while applying a penalty to heavily punish any misclassification of the rare malignant class.

2.3.3 Enhancing Model Robustness

Individual deep learning architectures frequently achieve remarkable accuracy, yet their generalization capabilities are often challenged by the limited diversity of public dermoscopic images. This lack of variety in the dataset can cause overfitting within the model, leading the network to memorize specific training patterns rather than learning robust visual features. Wu et al. provide a comprehensive overview of this limitation as well as other challenges inherent to the task of melanoma classification [24]. To address this issue and enhanced model robustness, various approaches have been developed, some of them are presented in this section.

First, data augmentation represents a widely adopted technique to enhance a model's generalization capability. This approach consists of artificially generating new training samples by applying various transformations to the existing images within the dataset. Perez et al. apply this strategy to melanoma classification by performing different types of variations on the dermoscopic images [25]. By increasing data diversity, this technique forces the network to learn invariant features, thereby significantly improving its ability to generalize to unseen data.

Next, an architectural strategy frequently employed to enhance generalization is Ensemble Learning. This approach consists of training multiple independent neural networks and aggregating their outputs typically through probability averaging to produce a single robust prediction. Harangi developed this methodology by combining distinct CNNs for skin lesion classification [26]. The benefit of this strategy lies in architectural complementarity: each model family captures different visual representations and their combinations tends to compensate for individual weaknesses while amplifying shared strengths. Gessert et al. illustrate this complementarity by bringing together several EfficientNet architectures trained across different image resolutions [27].

2.3.4 Multimodal Architectures

While early deep learning research in dermatology treated melanoma detection strictly as an image classification problem, in clinical practice, dermatologists must systematically consider the visual morphology of a lesion alongside patient-specific metadata such as age, biological sex and anatomical location. Recognizing that the accuracy of models based solely on imagery has limitations, the literature has progressively shifted towards Multimodal Deep Learning. These architectures are designed to simultaneously process images with clinical information. Recent studies have empirically proven that integrating patient metadata significantly boosts classification metrics. For instance, Pacheco and Krohling [28] demonstrated that adding clinical information reduces false positives by helping models discriminate between visually similar lesions in demographically distinct populations.

To integrate this metadata with a deep learning approach, researchers have explored various architectural strategies that can be organized in three distinct categories: early fusion, intermediate fusion, and late fusion [29]. Early fusion consists of combining the two types of data (image and metadata) before any data processing. However, this approach proves to be ineffective as it is constrained by the heterogeneity of the representation spaces: the volume of data from the pixels is disproportionate to the metadata vector which prevents the model from effectively learning the correlations between the two modalities. Next, the most frequent approach is the intermediate fusion also called feature-level fusion works as follows: each modality is independently processed to extract its different features. For instance, using a CNN for the image and an MLP for the metadata. These representations of the features are then merged together and fed into another model to produce the final prediction. Finally, the late fusion: each modality is processed from end to end by its own independent model. Then, their respective predictions or probabilities are aggregated by using a mathematical approach to generate the ultimate decision.

Currently, the architectures of multimodal models have evolved further and have shifted towards dynamic fusion strategies, heavily driven by attention mechanisms and Transformer architectures. Instead of simply concatenating the two types of data, attention-based frameworks use clinical metadata to modulate the visual network's focus during the feature extraction process. Furthermore, recent state-of-the-art approaches leverage Multimodal Transformers [30] where both image and clinical data are treated together as sequential tokens. Through the cross-attention mechanism, the clinical metadata actively guides the visual analysis by focusing on specific regions of the image. This dynamic interaction ensures that the visual features are always interpreted within the patient's demographic context. Despite

the encouraging results of these architectures, a major drawback must be considered: the cross-attention mechanism introduces a quadratic computational complexity $O(N^2)$, resulting in heavy memory footprints and highly demanding computational requirements. This can be a hindrance when working in environments where computational power is more limited.

2.4 Deployment of deep learning models

Training a high-performance deep learning model represents only the first step in the lifecycle of a medical artificial intelligence system. The real objective lies in deploying it and making the model accessible to end users (physicians, dermatologists or patients) via web or mobile interfaces that can be used in real-world conditions. This task known as 'model deployment' has attracted increasing attention in the literature in recent years.

2.4.1 Deployment architectures

In the literature, two main types of architecture are opposed in the deployment of deep learning models: server-side deployment and client-side deployment.

Server-side deployment via REST API

The most common approach involves encapsulating the trained model within a web service that exposes a REST API. The dermatoscopic image is sent from the client interface to a remote server which performs the inference and returns the result as a JSON response. Python frameworks such as Flask and FastAPI are widely used for this purpose. This architecture has the advantage of delegating the calculation to servers with substantial GPU resources, but it involves transferring patient data externally, raising significant privacy concerns.

Client-side deployment in the browser

In response to concerns about the confidentiality of medical data, an emerging trend involves performing inference directly within the user's web browser without transmitting the data to a third-party server. The release of TensorFlow.js, introduced by Smilkov et al. [31], marked a major step forward by providing a unified platform for training and inference directly in the browser with GPU acceleration, while also enabling full integration with Node.js for server-side deployment. Thanks to this library, it is possible to define, train and run machine learning models entirely within the browser on the client side, thereby ensuring data privacy. Goh et al. [32] present, in their review on the deployment of deep learning models in

client-side web applications, the state of the art in this field, notably focusing on the use of TensorFlow.js. They also emphasize the importance of using small deep learning models to optimize the front-end user experience, whether in terms of model loading or inference.

An example of this client-side approach and its associated benefits was demonstrated in a paper written by Dong et al. [33] .In this paper, the authors propose a solution for the deployment of AI-powered medical imaging. The proposed system utilises notably, TensorFlow.js, Open Neural Network Exchange (ONNX) which is an open format for representing and exchanging machine learning models and WebAssembly a binary format that enables high-performance code to be executed in web browsers alongside JavaScript. This solution ensures that patient data loaded into the application is neither transmitted over the network nor retained between sessions, thereby guaranteeing entirely local processing.

Chapter 3

Development of a deep learning model for melanoma

Now that the theoretical foundations have been established, it is time to delve into the heart of this thesis. This chapter aims to detail the architecture of the proposed model, following the chronological order of a deep learning model's development. First, the preprocessing stage will be covered, followed by a deeper look into the architecture of the multimodal model. Finally, the training strategy will be briefly discussed, particularly how class imbalance and optimization were handled.

Prior to detailing the development phase, it is essential to outline the architectural inspiration behind the model's design. The system presented is inspired by the work of Jaisakthi S. M. et al. [34]. This proposed model is not a direct replication but it adapts key defining characteristics most notably the use of the EfficientNet-B6 architecture and the integration of clinical metadata into the classification pipeline. Additionally, the dataset used to train this model is ISIC 2020, which contains both images and metadata that will be used to improve its performance.

3.1 Data Preparation

3.1.1 Preprocessing

Before training the deep learning model, both the dermoscopic images and their associated clinical metadata require careful preparation. For the image preprocessing phase, two distinct philosophies were evaluated. The minimalist approach applies only the essential transformations, limiting preprocessing strictly to resizing the image to the dimensions required by the network. Conversely, an interventionist approach leverages dedicated techniques designed to reduce visual noise and enhance

discriminative features. These techniques include hair removal—which detects and inpaints hair structures to reduce irrelevant texture occlusion—and intelligent cropping, which automatically localizes and centers the lesion to maximize diagnostically relevant content prior to resizing. Both strategies were tested within our pipeline. However, results indicated that these interventionist techniques yielded only a marginal impact on model performance. Consequently, to prioritize computational efficiency and architectural simplicity, the advanced image preprocessing steps were excluded in favor of the minimalist approach.

Beyond the visual inputs, the ISIC 2020 dataset provides patient-level clinical metadata, offering critical contextual information. Given the reasonable hypothesis that demographic and physiological factors correlate with the statistical likelihood of developing skin cancer, integrating these features can effectively supplement the visual data. The dataset includes three primary clinical variables: the patient's approximate age, biological sex, and the anatomical site of the lesion (categorized across seven distinct body regions).

To format these variables for neural network ingestion, several systematic preprocessing steps were applied. First, missing values in the continuous age variable were imputed using the dataset mean, followed by normalization to scale the distribution to an approximate range of 0 to 1. For the categorical variables (sex and anatomical site), missing entries were instead assigned to an explicit *unknown* class. Subsequently, one-hot encoding was applied to transform these textual attributes into binary indicator columns, a necessary step since neural networks can only process numerical inputs. This comprehensive preparation ultimately yields a formalized 11-dimensional numerical vector for each patient—comprising one dimension for normalized age, three for sex, and seven for the anatomical site—ready to be integrated into the multimodal architecture.

3.1.2 Data Augmentation

Training deep neural networks on finite medical datasets carries a substantial risk of overfitting. Rather than learning to generalize diagnostically relevant features, the model may instead memorize the specific characteristics of training images. To mitigate this risk, a stochastic data augmentation strategy was integrated directly into the training pipeline. The goal of this approach is to artificially increase the diversity of the training set by applying a set of randomized transformations to each image at every epoch ensuring that the model is never exposed to the exact same input twice. Leveraging such traditional data augmentation strategies is critical for improving model generalization [25].

The transformations performed are listed below and illustrated in Figure 3.1:

- **Spatial invariance:** Random horizontal and vertical flips, combined with rotations. These transformations compel the model to extract morphological features that are invariant to the acquisition angle.
- **Colorimetric perturbations:** A stochastic alteration of brightness and contrast is introduced into the pipeline. This modification simulates the illumination variability inherent to real-world clinical settings

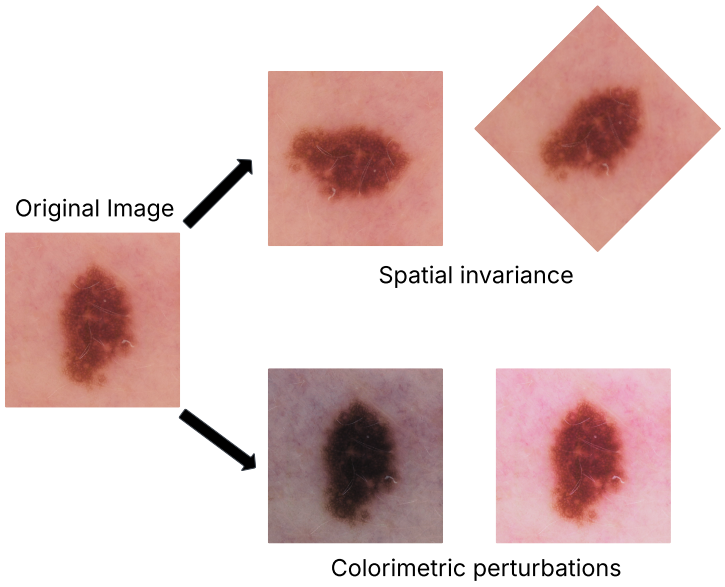


Figure 3.1: Data Augmentation Process

3.2 Model Architecture

The model developed in this study is based on a feature-fusion multimodal architecture. In other words, this model processes several types of data (in this case an image and clinical metadata) independently before combining the representations into a single one, just before the final classification decision is made. This approach is motivated by real-world diagnostic practice, in which the dermatologist does not limit themselves to a visual examination of the lesion but also incorporates contextual information such as age, gender or location to aid in the diagnostic decision-making process.

In terms of architecture, the model consists of two parallel and independent processing streams. The first is a CNN based on EfficientNet-B6 responsible for extracting a vector representation of the image. The second is a Multi-Layer Perceptron (MLP) designed to encode tabular metadata [35]. The representations produced by these two streams are then concatenated into a single vector which is fed to a linear classifier that produces a probability of belonging to the malignant class. The entire model is trained end-to-end allowing the two streams to adapt their representations jointly to the classification objective.

Figure 3.2 illustrates this general structure. The following sections describe each component of this architecture in detail.

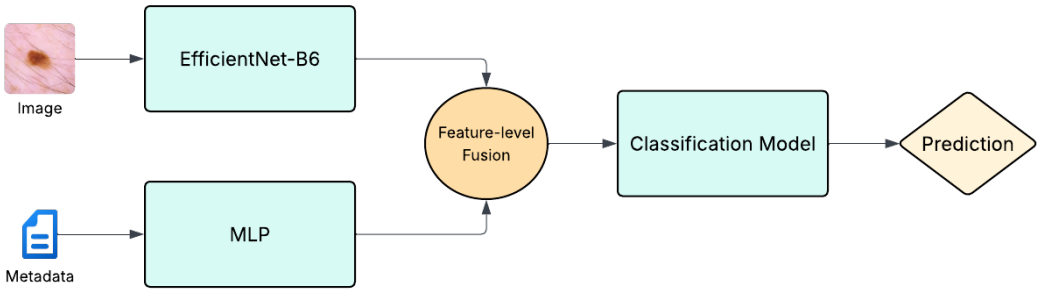


Figure 3.2: Global Architecture Overview

3.2.1 EfficientNet Architecture

The EfficientNet model family, introduced in 2019 by Mingxing Tan and Quoc V. Le. [4], marks a paradigm shift in neural network design. To grasp this innovation, it is essential to understand that the architecture of a CNN is governed by three fundamental dimensions :

- **Depth (d):** the number of successive computational layers
- **Width (w):** the number of filters within each individual layer
- **Resolution (r):** the spatial dimensions of the input image

Prior the advent of EfficientNet, improving model performance was typically achieved by scaling a single dimension at a time. However, this approach is limited: performance improvements progressively decreases while computational costs increase dramatically. The central insight of EfficientNet lies in demonstrating

that these three dimensions are interdependent and must therefore be scaled in concert. To address this limitation in a mathematically principled manner, the authors introduced the concept of Compound Scaling. The idea is to couple all three dimensions through a single global parameter, denoted ϕ, which acts as a unified scaling factor governing the overall size of the model. Accordingly, each dimension is defined as follows:

$$d = \alpha^{\phi}$$
$$w = \beta^{\phi}$$
$$r = \gamma^{\phi}$$
$$with \alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1$$

These variables are subject to the following constraint:

$$(\alpha \cdot \beta^2 \cdot \gamma^2)^{\phi} \approx 2^{\phi}$$

To understand this constraint, it is necessary to introduce the notions of FLOPS (Floating-Point Operations per Second), which quantifies the total number of mathematical operations the network must perform (represented by 2^{ϕ} in the constraint). Within a neural network, increasing the depth d raises the computational cost in a linear fashion. In contrast, doubling either the width w or the resolution r results in a fourfold increase in computation (hence the squared terms in the formula). The mathematical formulation thus enforces a straightforward property: each unit increase in the scaling factor ϕ by 1 doubles the total computational cost of the network. The constants α, β and γ determine how this computational budget is allocated across the three dimensions.

For this scaling strategy to be effective, it must be applied to a optimized baseline model. The EfficientNet family is therefore structured sequentially progressing from a common foundation towards increasingly large networks. EfficientNet-B0, the foundational architecture of the family, is the model upon which the constants α, β and γ are calibrated. The remainder of the family is then derived exclusively by incrementing the parameter ϕ, yielding models of progressively greater capacity. Specifically, EfficientNet-B0 corresponds to $\phi = 0$, and each subsequent variant (from B1 to B7) is generated by incrementing ϕ by one unit.

Finally, the efficiency of this model family stems from its use of specialized computational building blocks known as MBConv (Mobile Inverted Bottleneck Convolution). Rather than relying on computationally expensive standard convolutions, these blocks achieve their lightweight nature through two key concepts:

- **Depthwise Separable Convolutions:** Image analysis in a traditional network is similar to a single-block processing approach. The model attempts to simultaneously analyze the image geometry (spatial features) and its depth (channel-wise information) which requires massive computational power. The innovation of this method lies in splitting this process into two lighter stages. This separation of tasks allows the model to extract the same features while drastically reducing the computational load and memory requirements.

- **Squeeze-and-Excitation:** This is an attention mechanism. Initially the *squeeze* mechanism compresses the image to obtain a global overview then re-evaluates the importance of each channel through the *excitation* mechanism to amplify relevant features and suppress non-discriminative information. Consequently, the network dynamically learns to focus its attention precisely where the crucial information resides.

3.2.2 Visual Feature Extraction

The first flow of our multimodal architecture is designed to analyze the dermoscopic image in order to extract the spatial patterns, textures and color variations characteristic of skin lesions. The core architecture is based on the EfficientNet model family as presented above and more specifically on its variant EfficientNet-B6.

The limited size of the medical datasets exposes training such networks from scratch to a high risk of overfitting. To address this issue, Transfer Learning has been adopted. The network is initialized with weights pre-trained on ImageNet using the Noisy Student method [36]. This pre-training variant which involves injecting artificial noise such as dropout, stochastic depth and data augmentation, makes the feature extractor highly robust against perturbations.

Structurally, to use this network purely as a feature extractor, its original topology has been modified. The final classification layer, which was originally intended to predict the 1,000 ImageNet classes, has been disabled. As a result, when an image passes through the successive convolutional layers, it is directly condensed into a dense, one-dimensional vector of 2,304 features. This numerical vector acts as a comprehensive summary that encapsulates the entire visual semantics of the lesion, constituting a first element of the multimodal approach.

3.2.3 Clinical Metadata Encoding

Alongside the visual feature extractor, a second processing stream is dedicated to encoding clinical metadata. The input to this module is fed directly by the

11-dimensional vector from the preprocessing phase (combining the normalized age and the one-hot encoding of gender and anatomical site). This sub-network takes the form of a multi-layer perceptron (MLP) specifically designed to extract non-linear correlations from these tabular data.

The detailed architecture of this module is shown in Figure 3.3. As deep learning on such low-dimensional input data is particularly prone to overfitting, strict regularization mechanisms have been incorporated. The combined use of batch normalization and dropout forces the network to develop robust representations and prevents memorization of the training data.

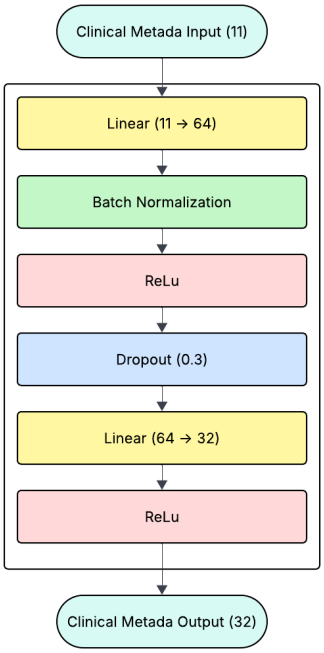


Figure 3.3: Architecture of the clinical metadata encoding network

At the end of this process, the 11 initial clinical variables are projected non-linearly to form a new dense vector comprising 32 latent features. This clinical vector is then ready to be combined with the visual vector produced by the image extractor.

3.2.4 Multimodal Fusion and Classification

The final decision flow of the architecture is based on a feature-level approach, the topology of which is summarized in the following diagram:

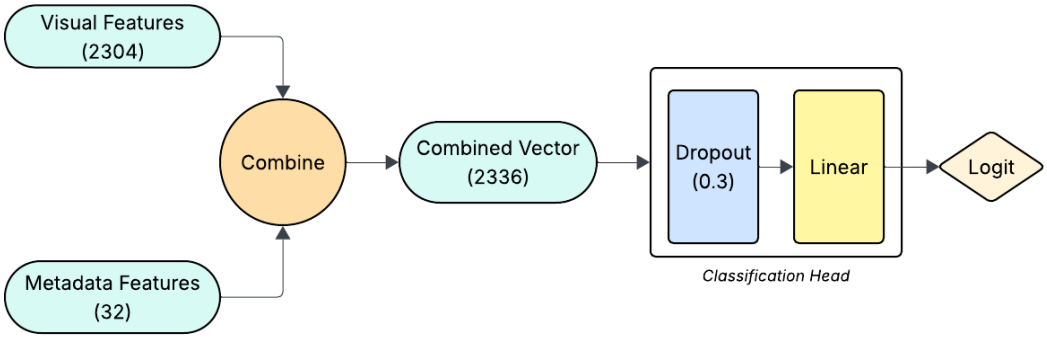


Figure 3.4: Architecture of the multimodal model fusion

As shown in Figure 3.4, this step directly merges the representations from the two data streams to form a single 2336-dimensional vector, which then passes through the classification head composed of a dropout layer and a linear layer. This strategy enables the classifier to learn jointly the correlations between the morphology of the lesion and the patient's clinical context, prior to the final diagnosis. This final result is a logit (value between $-\infty$ to $+\infty$) reflecting the probability of malignancy, thereby constituting the direct mathematical input of the weighted loss function during training.

3.3 Training Strategy

Once the architecture has been defined, the training phase requires a rigorous strategy to ensure the model converges whilst avoiding overfitting. The model training follows a standard protocol implementing a Stratified K-Fold cross-validation. This technique consists of partitioning the dataset into K distinct folds. while strictly preserving the initial proportion between the two classes within each fold. At each training iteration, the model is trained on $K-1$ folds and validated on the remaining fold. This operation is repeated K times to average the performance, thereby yielding more robust models and mitigating the bias of a potentially favorable validation split. Furthermore, an Early Stopping mechanism is deployed to prevent overfitting. It monitors the model results and halts the training loop when no improvement is observed for several consecutive epochs.

However, the core of this section focuses on the specific features of our system. First, handling the class imbalance is critical, as the training dataset is extremely unbalanced requiring specialized technique to mitigate this disproportion.Second a tailored optimization strategy is necessary to ensure the stable convergence of the learning process.

3.3.1 Handling Class Imbalance

One of the major challenges in the binary classification of melanoma is the significant class imbalance. If this imbalance is not addressed, the model will naturally seek to minimize its error by consistently predicting the majority class. Johnson and Khoshgoftaar extensively reviewed this phenomenon which produce an artificially high overall accuracy while masking a detection capability for the minority class that is almost zero [37].

To counter this, it is necessary to adjust the way in which the model measures its mistakes during the training. In Deep Learning, this error is tracked by a loss function, a mathematical metric that evaluates the gap between predictions and reality. Standard architectures typically rely on Binary Cross-Entropy (BCE) which calculates the average error for two-class problems. However, BCE treats all mistakes equally, regardless of class frequency or difficulty. To overcome this difference, Lin et al. introduced Focal Loss which adapts the standard BCE loss specifically for highly imbalanced datasets [38]. Instead of treating every error uniformly, Focal Loss dynamically reduces the importance of the majority cases. This forces the network to adapt its learning capacity to prefer hard cases. The formulation of Focal Loss is as follows:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

Where:

- p_t is the model's estimated probability for the target class
- γ is the focusing parameter. It scales the penalty based on the difficulty in classifying the target class.
- α_t is the weighting factor acting as a direct multiplier to give a priority to the minority class.

Mathematically, through our chosen configuration of $\alpha_t = 0.8$ and $\gamma = 2.0$, Focal Loss reduces the impact of the majority class. By minimizing the influence of benign samples, this tuning ensures that weight updates are driven by the rare melanoma cases.

3.3.2 Optimization

The optimization of a neural network is the mathematical and iterative process by which the model continuously adjusts its weights in order to minimize the loss function. This adjustment is governed by an algorithm known as the optimizer, which calculates the weight updates to be applied at each step.

In our architecture, the optimizer chosen is the AdamW algorithm (Adam with Weight Decay) [39]. Unlike the standard Adam optimizer, which links the L2 penalty (a mathematical constraint that prevents the network weights from becoming too large thereby avoiding overfitting) directly to the gradient calculation, AdamW decouples the regularization of the weights from the learning rate. This distinction enables the model to generalize more effectively and limits overfitting. The initial learning rate was set to 3×10^{-4} with a weight decay of 10^{-4}.

To dynamically adjust the amplitude of these updates over time, we have coupled the optimizer with a Cosine Annealing scheduler [40]. This technique adjusts the learning rate according to the following formula:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}}\pi\right)\right)$$

where $\eta_{\min}$ and $\eta_{\max}$ denote the lower and upper bounds of the learning rate, T_{cur} the current number of epochs, and $T_{\max}$ the total number of epochs. The learning rate thus decreases continuously and monotonically along a cosine curve which produces two complementary effects during training.

At the start of training, the learning rate remains relatively high allowing the model to explore the parameter space more freely, escape from local minima and reach promising regions of this non-convex space. Towards the end of training, the learning rate gradually decreases to very low values, enabling a finer adjustment of the weights and thus promoting stable convergence towards a high-quality local minimum.

Chapter 4

Model Deployment and Web Integration

The aim of this chapter is to translate the PyTorch model developed earlier into an accessible and functional web application. In a medical context where data confidentiality is essential, the chosen approach relies on execution entirely on the client side directly within the browser, enabled by WebAssembly and ONNX Runtime technologies. This architectural choice offers a double advantage: images of lesions never leave the user's device thereby eliminating any risk of data leakage, whilst avoiding the costs and constraints associated with server infrastructure. The following sections describe the entire deployment pipeline, from exporting the model in ONNX format to its integration into an interactive interface enabling real-time inference.

Figure 4.1 illustrates the global architecture of the application. As depicted, the system is structured around three main components.

1. The conversion of the PyTorch model into an ONNX model using a Python script.

2. The development of the application's front-end using the TypeScript programming language, which is characterized by strong typing. This part handles the application logic and the orchestration of data and processes.

3. The local inference managed by WebAssembly coupled with ONNX Runtime.

The following sections will detail each of these components, demonstrating their utility and how they work within the application.

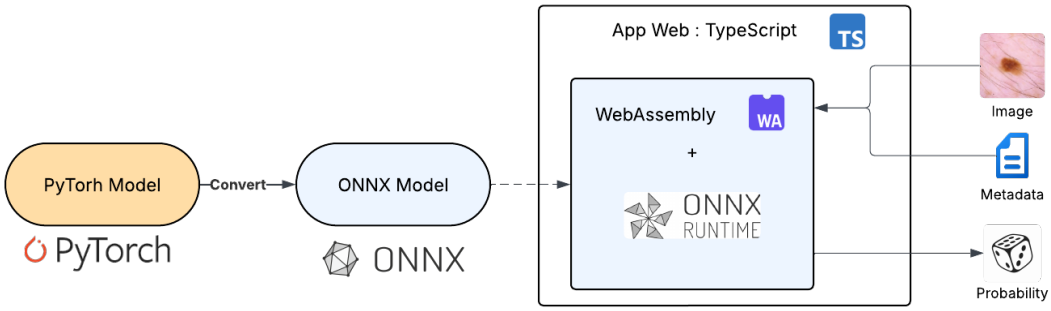


Figure 4.1: Global Architecture of the Web Application

4.1 Export with ONNX

4.1.1 Introduction to ONNX

ONNX (Open Neural Network Exchange)[6] is an open and standardized format designed to represent deep learning models. Developed by Microsoft and Facebook (now Meta), ONNX was built to address one of the key problems in deep learning: the lack of interoperability between frameworks. Its role is to provide a common representation of neural network computation graphs.

ONNX is built around a DAG (Directed Acyclic Graph) where each node represents a mathematical operator (convolution, normalization, activation, etc.) and edges carry the tensors flowing between them. This graph is serialized using Google's Protocol Buffers format which provides a compact and portable representation that is independent of the programming language.

The value of this standard for model embedding is based on two advantages. The first is interoperability, one of the key benefits of ONNX is that it decouples the training phase of a model from its deployment phase. A model trained in PyTorch, TensorFlow, or any other major framework can be exported as a *.onnx* file and loaded in any environment with a compatible runtime. The reference runtime, ONNX Runtime (ORT) [8], is specifically optimized for inference workloads.
The second advantage is inference optimization. ONNX comes with a set of graph optimization tools that reduce latency and memory usage at inference time. These include operator fusion, redundant node elimination and more.

In short, ONNX acts as a *'universal language'* that converts deep models into lightweight files ready to be interpreted by web or mobile environments.

4.1.2 Export Process

Converting a PyTorch model to the ONNX standard involves a structural paradigm shift: moving from a dynamic computation graph (where mathematical pathways are built on-the-fly during execution) to a static execution graph (where the entire network is pre-compiled into a fixed and self-contained sequence).
Before even performing this mathematical translation and according to the official PyTorch documentation[1], a strict preparatory step is required: switching the model to evaluation mode. This instruction is mandatory because it freezes the network's internal state and disables the stochastic behaviors specific to the training phase such as Dropout. Without this precaution, the exported model would retain non-deterministic behavior making any inference non-deterministic.

Once the model is stabilized, the conversion is carried out through a mechanism called Tracing. Since the ONNX exporter cannot directly interpret the Python source code, it must 'observe' the network in action to infer its architecture. The process therefore consists of feeding the model with dummy inputs that mimic the format of real data. By simulating a complete forward pass with this data, the exporter tracks the flow of tensors and meticulously records, in a sequential manner, every mathematical operation encountered (convolutions, matrix multiplications, activation functions). It is the frozen trace of these operations that forms the backbone of the final static graph.

However, the complexity of our architecture required custom parametrization of this export process due to its multimodal nature. Unlike a standard neural network processing a single matrix, our model performs a fusion between two different data streams: the visual features extracted by the EfficientNet backbone on one hand, and the patient's clinical data processed by our sub-network on the other. Concretely, the exporter had to be configured to explicitly name each input node of the model (e.g., *image_input* and *meta_input*). This labeling step provides the ONNX model a clear interface, which will be essential for the web integration. Consequently, the execution engine does not blindly feed tensors into the model: it knows with absolute certainty into which specific input node to inject the pixel matrix and into which one to pass the patient's clinical data vector.

4.1.3 Encapsulation of preprocessing

Software integration of a computer vision model requires absolute rigor: the images used for inference must undergo mathematical preprocessing that is strictly identical to that applied during the training phase. Initially, the architectural

[1]PyTorch Documentation: `https://docs.pytorch.org/docs/2.12/index.html`

approach adopted involved delegating this step to the web application, with resizing and normalization calculations performed on the client side. However, this approach quickly revealed a technical vulnerability inherent to this environment. The JavaScript ecosystem used to develop the web application offers options for image processing, but remains less mature and less specialized than environments such as Python for low-level matrix operations. Consequently, subtle variations in the precision of the calculations led to differences in the inference results. In practice, the same image analyzed by the local environment (PyTorch) and by the web application (JavaScript coupled with ONNX) resulted in slightly different predictions. In the context of medical diagnosis, such inconsistency in results is unacceptable, thereby necessitating a review of the pre-treatment strategy.

To resolve this inconsistency, an architectural decision was made to embed the preprocessing logic directly within the ONNX model. Instead of delegating this task to the web client, a dedicated preprocessing wrapper class was developed during the exportation. This layer acts as a standardized staging area for data preparation. This involves resizing, normalizing and standardizing the images. During export, these instructions were translated and encoded as mathematical nodes within the ONNX file itself.

4.2 Web Application

This section details the design of the web application which acts as the direct interface between the ONNX model and the end user. The main objective during its development was to design a digital tool that is intentionally simple and intuitive. Rather than offering a multitude of features, the interface has been designed to focus strictly on its core clinical purpose: the detection of melanoma.

4.2.1 Technologies and Application Logic

The client interface is developed in TypeScript, a strongly typed programming language that helps prevent potential runtime errors. Before carrying out the inference (described in the next section), the front-end component handles the preprocessing of these two data streams:

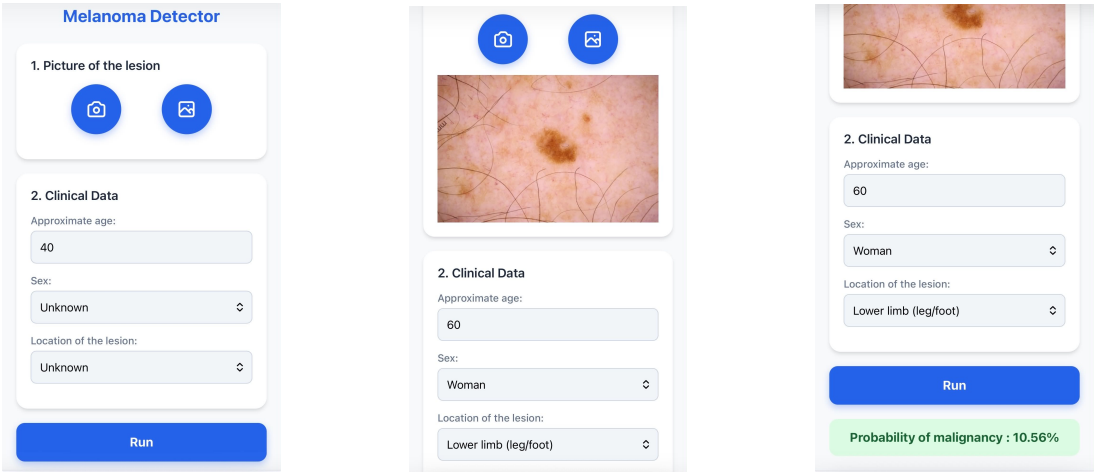
- **Image processing:** The uploaded image is read by the browser to extract the raw values for each pixel (the red, green and blue channels). This visual grid is then converted and flattened into a one-dimensional mathematical vector in accordance with the strict format required by the ONNX standard.

- **Metadata processing:** At the same time, intercepting the DOM (Document Object Model) allows us to capture the clinical variables entered into the form by the user. This data is then processed using a local mathematical encoding to generate the second tensor required by the model.

4.2.2 Clinical Interface and User Workflow

As soon as the web application is accessed, the ONNX model is loaded asynchronously into memory. As shown in Figure 4.2, the user is then presented with a home screen that allows them to submit images of lesions and clinical data. Firstly, the user can upload an image, either via the device's camera or by importing it from the gallery. The image capture process is then completed by entering clinical parameters (age, gender, anatomical location), which will feed into the model's second data stream during inference. Once this information has been entered, the user initiates the local analysis using a button at the bottom of the page provided for this purpose.

After the analysis, the interface displays the probability that the lesion is malignant. To simplify the interpretation of the result, this value is associated with a visual indicator (Figure 4.2c): the interface displays green for a benign prediction or red if the calculated probability exceeds the pre-set critical threshold.



(a) Home Page (b) Add the Data (c) Diagnosis

Figure 4.2: Overview of the screens in the 'Melanoma Detector' app

4.3 Inference via WebAssembly

4.3.1 Introduction to WebAssembly

WebAssembly (Wasm) is a standard of the World Wide Web Consortium (W3C) [7]. Nowadays, it is considered as the fourth fundamental language of the web, alongside HTML, CSS and JavaScript. However, Wasm is not a programming language but a low-level binary instruction format. It was created to enable the compilation of certain higher-level languages such as C and C++.

The core principle of WebAssembly is the compilation of source code into a .wasm binary module. This module is then executed within a virtual machine. Before execution, the code is statically validated to prevent undefined behavior. Runtime engines, such as web browsers, then translate the bytecode into native machine code. Historically, web applications relied on JavaScript to execute their logic. But this approach can have its limitations, particularly in the field of AI. WebAssembly, on the other hand, allows complex code to be executed directly from the browser.

WebAssembly offers several notable technical advantages, which are briefly explained below. Firstly, isolation: modules run in a sandboxed environment, meaning they have no direct access to the OS. Any interaction with the outside world (for example with files) must go through an interface explicitly exposed by the host. Finally, portability: WebAssembly allows the same module to be executed on any platform with a compatible runtime, regardless of the operating system or hardware architecture.

In summary, WebAssembly offers a high-performance, secure and portable solution for running advanced code, such as deep learning models inside a web environment. It overcomes the limitations of JavaScript thanks to an optimized binary format.

4.3.2 Running the Model on the Web Application

In our melanoma detection application, WebAssembly acts as the system's core computational engine, bridging the performance gap between the web browser and the requirements of deep learning. The execution of this inference is managed by the ONNX Runtime Web library. When the client application is initialized, the ONNX file representing the model is loaded into memory. It is at this crucial stage that the software architecture explicitly mandates the use of WebAssembly as the computing environment. This ensures that the computation graph is not interpreted by the browser's JavaScript engine, but delegated to the Wasm binary module for execution at near-native speed.

More specifically, once the image has been serialized and the metadata encoded by the TypeScript application layer, these two tensors are passed to the WebAssembly module. This module handles the millions of computationally intensive matrix operations (convolutions, mergers, activation functions). Thanks to its low-level instruction format, WebAssembly executes the entire ONNX graph by making optimal use of the client device's processor, before outputting the malignancy score to the graphical user interface.

Chapter 5

Results

This chapter evaluates the proposed melanoma classification system across two main phases. After defining the key performance metrics, the first phase analyzes the deep learning model's predictive performance, training dynamics, and clinical threshold optimization, alongside a benchmark against state-of-the-art architectures. Finally, the second phase assesses the client-side web deployment, evaluating ONNX model parity, load times, and inference latency to quantify the trade-off between local execution constraints and clinical data privacy.

5.1 Evaluation metrics

To evaluate model performance in the field of deep learning, specific metrics are employed to assess the model's predictions. Before delving further into the analysis of the results obtained, it is essential to clarify the key performance metrics that will be used throughout this report.

5.1.1 AUC

The AUC (Area Under the Curve) is used to assess the ability of a binary classification model to distinguish between two classes. This measure is associated with the ROC (Receiver Operating Characteristic) curve, which graphically represents the trade-off between the true positive rate and the false positive rate as the classification threshold is varied. The AUC corresponds to the area under this ROC curve and takes a value between 0 and 1. The closer the AUC is to 1, the better the model performs. Conversely, a value of 0.5 corresponds to a model with no predictive power, equivalent to a random choice.

One of the main advantages of AUC is that it is independent of the chosen

decision threshold, making it particularly useful when classes are highly imbalanced as is the case with our ISIC 2020 dataset.

5.1.2 Accuracy

Accuracy represents the overall proportion of correct predictions made by the model, encompassing both true positives and true negatives. Mathematically, it is defined as the ratio of all correct assessments to the total number of cases examined:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

While it provides a general overview of the model's performance, accuracy can be highly misleading when evaluating severely imbalanced datasets. In such contexts, a model could achieve a high accuracy score simply by systematically predicting the majority class while completely failing to identify the rare malignant cases. Therefore, it should always be interpreted alongside metrics like AUC and recall.

5.1.3 Recall

The recall, also known as sensitivity evaluates the ability of a classification model to correctly identify all real positive cases. Mathematically, it's defined as the ratio of true positives (TP) divided by the sum of true positives and false negatives (FN):

$$Recall = \frac{TP}{TP + FN}$$

The closer this value is to 1, the better the model is at identifying truly positive cases without omitting any. In the field of medical imaging, recall is a crucial metric, its goal is to minimize the number of false negatives. In dermatology, the objective is to avoid incorrectly diagnosing a malignant lesion as benign. Moreover, it is clinically preferable to generate false positives rather than miss a cancer diagnosis.

5.1.4 Precision

The precision measures the accuracy of the model's predictions when it detects a positive case. Mathematically, it's defined as the ratio of true positives (TP) divided by the sum of true positives and false positives (FP):

$$Precision = \frac{TP}{TP + FP}$$

The closer this is value is to 1, the fewer false positives the system generates.

5.1.5 F1-Score

The F1-score, also known as the F-measure, represents the harmonic mean of precision and recall. Mathematically, it is defined by the following formula:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The purpose of this metric is to provide a single indicator that summarizes the overall balance of a model.

5.2 Model Analysis

5.2.1 Experimental Setup

Prior to detailing the model training process and its results, it is essential to outline the experimental setup. The training was conducted on the Lyra CÉCI HPC cluster[1]. The hardware specifications utilized for this work are summarized in Table 5.1.

Component	**Specification**
CPU	32-core @ 3.80 GHz
RAM	128 GB DDR5 ECC @ 4.8 GHz
GPU	NVIDIA RTX 6000 ADA (48 GB GDDR6 ECC)

Table 5.1: Hardware specifications used for model training

5.2.2 Analysis of Model Learning

Following the definition of the experimental setup, we now analyze the model's training dynamics to verify its capacity for generalization over memorization. To illustrate this process, we will analyze the learning of the third cross-validation fold chosen for its performance and representativeness.

Figures 5.1 and 5.2 illustrate the model's learning dynamics through the evolution of the loss function and the AUC. During the first five epochs, rapid learning is observed with a sharp decrease in the training loss and a corresponding increase in training AUC. The model's generalization capacity continues to improve at a moderate pace until epoch 14, where the validation AUC peaks at 0.912. Beyond this point, the validation loss begins to increase while the training loss continues

[1]Lyra cluster: `https://hpc.ulb.be/lyra.php`

to drop, marking the onset of overfitting. This divergence demonstrates that the model is starting to memorize the training data rather than extracting generalizable features. This confirms the critical role of our early stopping mechanism, which successfully halted the training process (after 10 epochs without improvements) and preserved the optimal model parameters at epoch 14, ensuring the robustness of the final classification results.

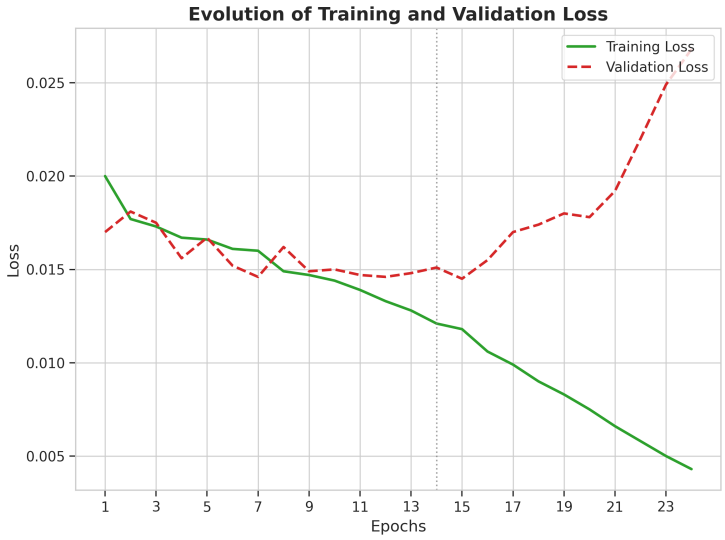


Figure 5.1: Evolution of the loss over epochs

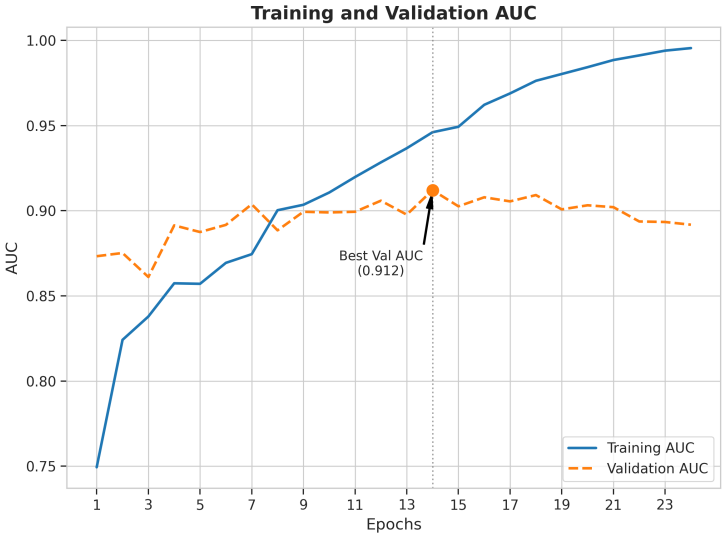


Figure 5.2: Evolution of the AUC over epochs

5.2.3 Performance Metrics

The final tested predictive performance of the proposed architecture is summarized in the Table 5.2. Furthermore, since certain metrics (such as accuracy, recall and precision) are threshold-dependent, the selection of an appropriate decision boundary is crucial. Given the severe class imbalance within the dataset, a default threshold of 0.5 would be highly inappropriate. Consequently, the utilized threshold was optimized to prioritize the detection of malignant lesions. This choice and its corresponding results are detailed further in Section 5.2.5.

Metric	**Value**
AUC	0.910
Accuracy	0.855
Recall	0.807
Precision	0.092
F1-Score	0.165
Threshold	0.145

Table 5.2: Results of the performance metrics for the developed model

5.2.4 State-of-the-Art Comparison

To fully contextualize the performance of our developed multimodal architecture, it is essential to benchmark our results against established models in recent literature. As the ISIC Archive is one of the most frequently used repositories for automated melanoma detection, numerous deep learning approaches have been validated on its official datasets. For this comparison, we selected a subset of widely cited and recent studies evaluated on the ISIC Archive benchmarks. These references represent various modern approaches to addressing the classification of melanoma.

Table 5.3 summarizes the performance of these reference models alongside own work, using the AUC as evaluation metric.

Paper	Model Architecture	AUC
Classification of skin cancer from dermoscopic images using deep neural network architectures [34]	EfficientNet-B6 Ranger optimizer	0.9681
Identifying Melanoma Images using EfficientNet Ensemble: Winning Solution to the SIIM-ISIC Melanoma Classification Challenge [41]	EfficientNet Ensemble	0.9490
Melanoma classification from dermatoscopy images using knowledge distillation for highly imbalanced data [42]	EfficientNet Teacher-Student Framework	0.9295
Automatic Malignant and Benign Skin Cancer Classification Using a Hybrid Deep Learning Approach [43]	Xception	0.917
Our Contributions	EfficientNet-B6 Multimodal model Focal Loss	0.914
Melanoma and Nevus Skin Lesion Classification Using Handcraft and Deep Learning Feature Fusion via Mutual Information Measures [44]	MobileNetv2	0.8964

Table 5.3: Comparison of skin cancer classification models

As indicated by the quantitative results, our model's performance sits slightly below the average baseline established by the top-tier models in recent literature. Although it does not rank at the top of the overall rankings, the AUC score obtained remains quite acceptable and far from negligible, demonstrating solid and clinically viable diagnostic capability.

5.2.5 Results and Discussion

The proposed multimodal architecture demonstrates strong discriminative capabilities, evidenced by a mean AUC of 0.910. As a threshold-independent metric, this score confirms that the network has deeply internalized the morphological features of melanoma.

While the AUC highlights the model's intrinsic predictive power, discrete metrics such as accuracy, recall, and precision are tied to a specific threshold. In the context of melanoma, minimizing false negatives is a clinical imperative because missing

a malignant lesion corresponds to ignoring a potentially deadly melanoma and so is far more critical than the risk of a false positive, which results in a simple precautionary consultation. Consequently, the decision threshold was deliberately calibrated along the Precision-Recall curve (see Figure 5.3) to guarantee a minimum recall of 0.80, ensuring that malignant cases are detected as an absolute priority. The resulting threshold is therefore 0.145, which yields a high recall of 0.807.
Then, given the extreme class imbalance of the ISIC 2020 dataset, enforcing this high recall forces the model to broaden its positive predictions and inevitably generating a significant volume of false positives. As a direct result, precision drops to 0.092, subsequently pulling the F1-score down to 0.165. Finally, the overall accuracy is reduced to 0.85, penalized by misclassifications errors in the majority class.

Ultimately, the results are visually confirmed by the confusion matrix (Figure 5.4). This shows that the large majority of malignant cases are correctly detected with only a small number of false negatives. Although false positives account for a significant proportion, they remain a clinically acceptable trade-off for a strategy deliberately designed to maximize recall.

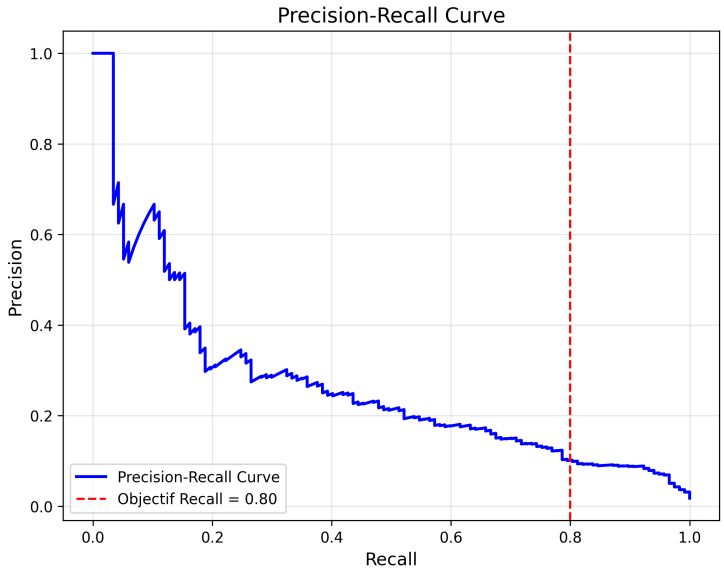


Figure 5.3: Precision-Recall Curve

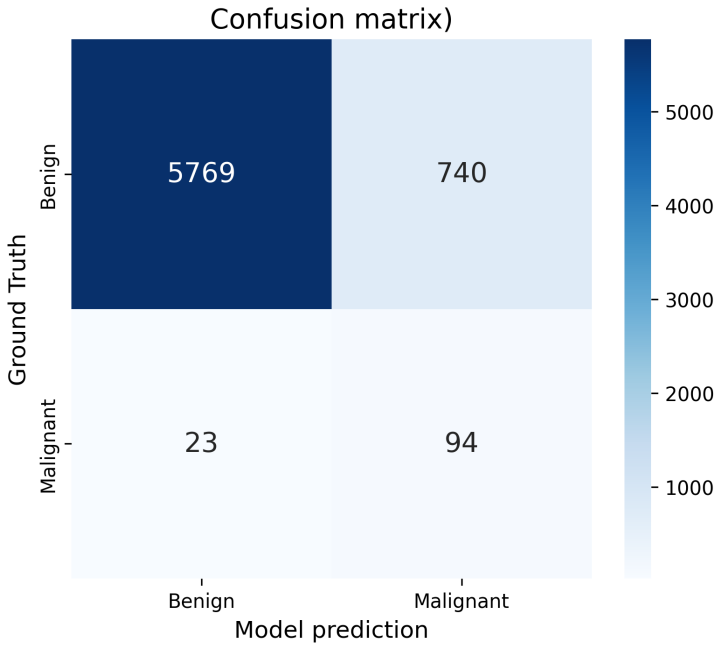


Figure 5.4: Model Confusion Matrix

5.3 Web Application Analysis

Deploying a deep neural network such as EfficientNet directly within the user's browser represents a major technical challenge for client-side execution. This approach offers the dual advantage of ensuring clinical data confidentiality and remaining independent of a stable internet connection once the model is loaded. However, shifting the computational load to the local machine raises important questions regarding the reliability and usability of the system. The objective of this section is to assess the viability of this deployment via ONNX Runtime and WebAssembly by analyzing three fundamental criteria: the mathematical fidelity of the converted model, the weight of the model, and the real-world inference latency.

5.3.1 Experimental Conditions

Before proceeding with the evaluation of the web application, it is essential to define the experimental conditions. The hardware specifications and the software environment are summarized in Tables 5.4.

Component	Specification
Device	iPhone 13
OS	iOS 26.5
CPU	Apple A15 Bionic (6-core)
RAM	4 GB
ONNX Runtime Web	v1.24.3
Web Browser	Firefox 151.1
Network Speed[1]	207 Mbps

[1] Speed test performed via `https://www.speedtest.net/`

Table 5.4: Experimental Conditions for the Web Application Evaluation

5.3.2 Model Parity

The first step in validating our application is to ensure that converting the model from its original PyTorch format to the ONNX standard has not compromised its predictive capabilities. Specifically, the objective is to verify that the format transition does not introduce any mathematical divergence from the original predictions.

To quantify this variation, a parity test has been implemented. A subset of 100 dermoscopic images, along with their clinical metadata, was randomly selected. Each image was processed using the original PyTorch model running locally, as well as the ONNX model deployed within the web application. The resulting malignancy probabilities were recorded for each prediction, allowing us to calculate the average and maximum differences between the two environments.

To demonstrate that the remaining differences stem from the application's runtime environment rather than from errors in the model's internal mathematical calculations, the tests were conducted in parallel on two separate web browsers. Table 5.5 shows the results of this experiment. These results prove that the conversion was highly successful and that the PyTorch and ONNX models share the exact same mathematical architecture. The mean variance remains below 0.2% in the worst-case scenario (Google Chrome), which is clinically negligible for a binary classification task.

Browser	Average Probability Difference	Max Difference
Google Chrome	0,0018 (0,18 %)	0,0161 (1,61 %)
Mozilla Firefox	0,0001 (0,01 %)	0,0013 (0,13 %)

Table 5.5: Results of the parity test between the PyTorch model and the ONNX model, performed on two different browsers

5.3.3 Loading of the Model

The ONNX model loading process within our application represents a critical phase, as it dictates the user's first interaction with the system. It should first be noted that the exported model has a substantial file size of 156 MB. To evaluate the time required to load the model into the web environment, a series of tests was conducted, with the page being refreshed between each iteration to simulate a cold start. As illustrated in Figure 5.5, the results indicate that the loading time remains largely stable across different runs, yielding an average duration of 17.39 seconds. The minor variations observed between measurements can be readily attributed to natural fluctuations in network bandwidth.

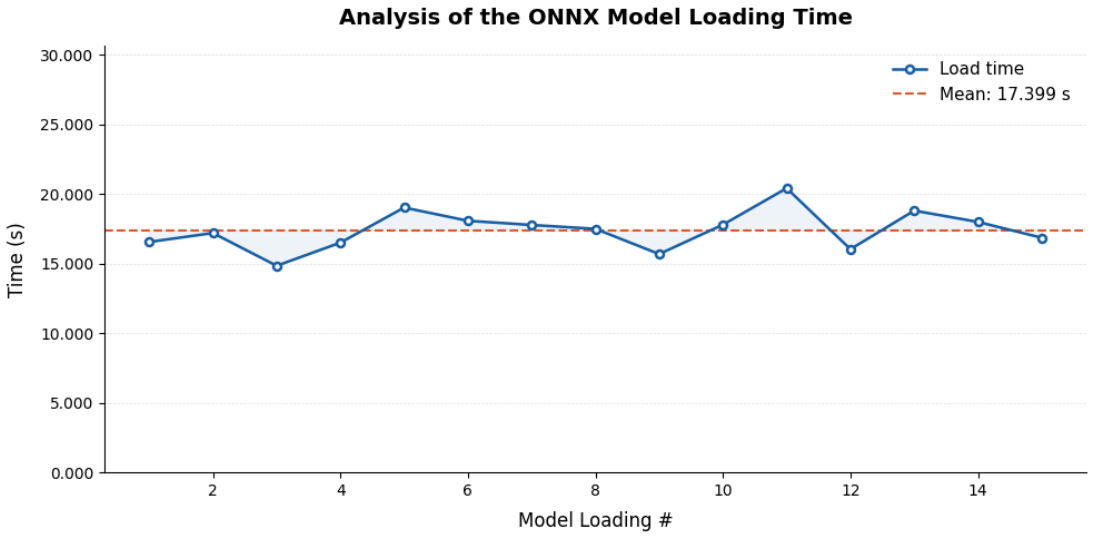


Figure 5.5: Analysis of the ONNX Model Loading Time

5.3.4 Inference Latency Analysis

The inference time analysis is an essential element for measuring the usability of the application. Figure 5.6 compares the latency, expressed in milliseconds on a logarithmic scale, between the original PyTorch model running locally and the

ONNX model running client-side on a sequential sample of 100 images.

The major observation from this test lies in the performance gap between the two approaches. Benefiting from low-level optimizations, the native PyTorch model processes each image in approximately 150 ms and maintains remarkable inference stability throughout the sequence. In contrast, the web-deployed ONNX model requires between 2,000 and 5,000 ms to accomplish the same task. Beyond this initial delay, the ONNX configuration also demonstrates a progressive latency drift towards the 5,000 ms mark. This continuous performance degradation is primarily driven by the physical hardware limitations of the mobile device under sustained load.

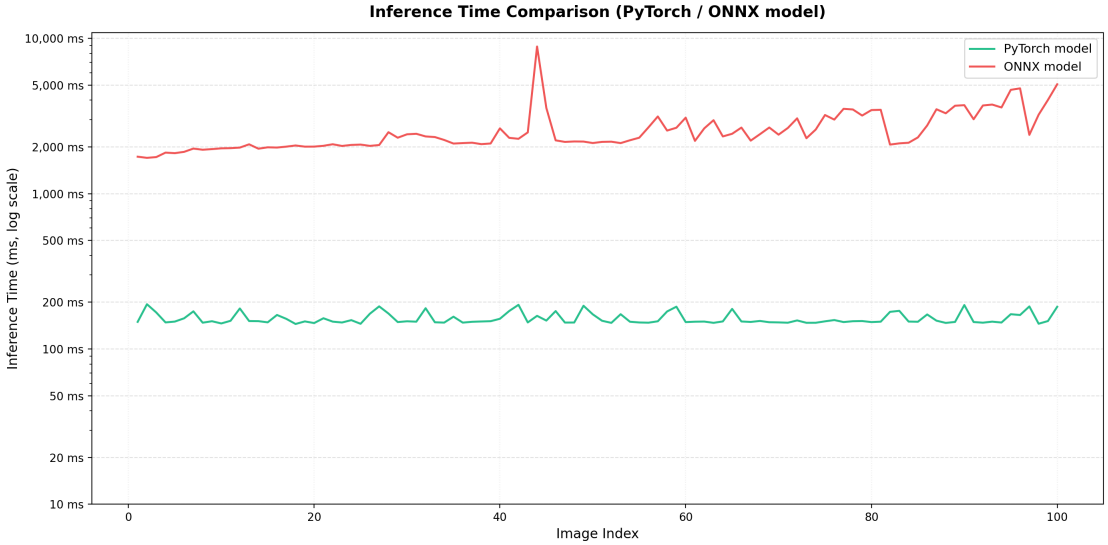


Figure 5.6: Inference latency comparison between the native PyTorch model and the ONNX model

5.3.5 Results and Discussion

After carrying out various experiments on the deployment of our model inside a web application, analyses of these results are performed to draw the necessary conclusions. The goal is to determine the viability of the client-side approach for developing a tool of this type. Prior to analyzing performance, the parity test validates the mathematical integrity of the converted model. The recorded micro-variations further suggest that inference execution is influenced by browser-specific runtime environments, which introduce an abstraction layer that lies outside the developer's direct control.

Subsequently, the analysis of the model loading process demonstrated highly stable performance, yielding an average initialization time of 17.39 seconds. While this loading time may be considered acceptable in a clinical application context, particularly given the advantage of complete independence from network stability once loaded. It nonetheless highlights a clear limitation of the current solution, underscoring the need for future optimization.
Then, the inference latency analysis revealed a substantial performance gap between the native PyTorch model and the ONNX model executed via WebAssembly and ONNX Runtime. While the PyTorch model processed images in approximately 150 ms, the ONNX model required between 2,000 and 5,000 ms. This disparity is largely attributable to the inherent differences in available hardware resources between the two execution environments. Furthermore, a progressive increase in inference time was observed with the ONNX model. One possible hypothesis for this latency drift is hardware resource saturation. Ultimately, this execution time remains acceptable for a clinical web application, particularly since users are unlikely to perform continuous inferences in a real-world clinical workflow.

In conclusion, despite the performance limitations inherent to the application's client-side approach, these drawbacks remain highly acceptable when weighed against the advantages it provides. These performance costs should be viewed as a necessary trade-off to achieve three major advantages: enhanced user privacy, reduced backend infrastructure costs, and complete network independence once the initial model is loaded.

Chapter 6

Conclusion

6.1 Key Contributions

The objective of this work was to develop a comprehensive melanoma detection system encompassing both the design of an advanced deep learning model and the implementation of a mobile-oriented web application. A distinctive feature of the proposed system is its fully client-side architecture, which addresses the key limitations inherent to server-side approaches, particularly network dependency and privacy concerns associated with the transmission of sensitive medical data. Furthermore, the system is released as an open-source project, thereby contributing to addressing a notable gap in the availability of complete and publicly accessible solutions in this domain.

The first component of this work focuses on the development of a deep learning model capable of classifying skin lesions as either malignant or benign. To train this model, the ISIC 2020 dataset was employed, a benchmark widely adopted in the field of dermoscopic image analysis. A notable characteristic of this dataset is that it provides both dermoscopic images and clinical metadata relating to patient information. The incorporation of such metadata into the model, forming a multimodal architecture, is a well-established technique in the domain and was therefore adopted in the present study.
The proposed model is structured around two complementary branches. The first consists of a visual feature extractor built upon an EfficientNet-B6 backbone, which produces a 2304-dimensional feature vector encoding the visual representations derived from the input image. The second branch comprises a metadata encoder implemented as a Multi-Layer Perceptron (MLP), designed to encode clinical attributes such as patient age, sex, and the anatomical site of the lesion, yielding a compact 32-dimensional embedding vector. These two representations are subse-

quently combined through a feature-level fusion strategy, in which the visual and metadata embeddings are merged into a unified representation. This fused vector is then passed through a classification head, which produces the final diagnostic output.

The second part of this work addresses the software engineering challenges involved in deploying the PyTorch model into a web application accessible to end users. As a first step, the trained network was converted to the ONNX format which provides a standardized representation of the model. To mitigate the numerical precision discrepancies observed between Python and JavaScript execution environments, the image preprocessing and normalization logic was directly encapsulated as mathematical nodes within the ONNX graph at export time, ensuring consistent inference behavior across platforms. In parallel, a mobile-oriented web application was developed in TypeScript to handle the full application logic. Model inference is performed client-side through the ONNX Runtime Web library. To overcome the computational limitations inherent to traditional JavaScript engines, the execution of the model's intensive operations is delegated to WebAssembly. The integration of this binary format significantly increases the computational capabilities of the web application directly within the browser.

6.2 Results and Discussion

The analysis of the results obtained throughout this work demonstrates the viability of the proposed end-to-end system. The following discussion reviews the performance of each individual component in turn, examining both the diagnostic accuracy of the deep learning model and the efficiency of its local execution within the web application interface.

First, following a training process that proceeded in a stable and well-converged manner, the proposed architecture demonstrated strong intrinsic discriminative capacity, as reflected by a mean AUC of 0.910. Placed in the context of the existing literature, this performance falls slightly below that of the most advanced architectures proposed in the state of the art. Regarding threshold-dependent metrics, a deliberate clinical trade-off was made in the selection of the decision threshold. Given the critical medical imperative to minimize false negatives in melanoma classification, the threshold was mathematically optimized to ensure a high recall of 0.807. As a direct consequence of this increased recall, the model was mechanically constrained to broaden its positive predictions, resulting in a substantially increased false positive rate. This inevitably led to a significant drop in precision, which fell to 0.092, which in turn negatively affected the overall accuracy (0.855) and the

F1-score (0.165). Nonetheless, these scores must be put into clinical perspective: a high false-positive rate leading to a simple precautionary consultation is a widely accepted and safe compromise compared to the catastrophic risk of a false negative.

Afterwards, the evaluation of the web application begins with a verification of the integrity of the model conversion process. To this end, a parity check was conducted consisting of performing identical inference tests on both the original PyTorch model and the exported ONNX model running within the web application. The results confirmed the success of the conversion, revealing a maximum numerical deviation of less than 0.2% in the worst-case scenario, with marginal variability depending on the browser environment. These micro-variations, attributable solely to floating-point handling differences across browser execution environments, are considered negligible and demonstrate that both models maintain equivalent mathematical fidelity.
Subsequently, an inference performance analysis was carried out, shedding light on the expected latency gap between native and browser-based execution. While a single inference on the local PyTorch model completes in approximately 150 ms, the same operation within the web application requires between 4 and 5 seconds. A substantial difference that nonetheless remains entirely acceptable in the context of a clinical web application. Despite this performance gap, both execution environments demonstrate consistent inference stability across repeated runs, suggesting that the deployment pipeline does not introduce any systematic degradation.

In conclusion, these results validate the proposed system. The deep learning model demonstrates clinically acceptable performance while the web application successfully executes inference entirely client-side, with satisfactory mathematical fidelity and execution times. The client-side approach therefore proves fully viable for this class of problem, ultimately yielding an operational system that combines predictive performance with network independence.

6.3 Future Work

This work has resulted in a first functional prototype of the proposed system. Nevertheless, numerous opportunities for improvement remain to be addressed in order to enhance both the practicality and robustness of the tool. This prototype constitutes a solid foundation upon which several complementary technical developments could be pursued, with the aim of progressively aligning the system with the demands of real-world clinical deployment.These potential improvements focus on three main areas: architectural optimization, advancement of the web integration, and the management of image capture conditions.

From a model engineering perspective, a particularly relevant improvement would concern the compression of the architecture through model quantization. Han et al. notably describe this compression technique as a means of optimizing the deployment of deep neural network architectures [45]. Applying this approach directly to the ONNX file would shrink the total model size, significantly reducing the initial loading time when the application is first opened in the browser. Beyond this storage gain, the simplification of the mathematical operations within the computational graph would additionally reduce the processing load on the user's device, directly accelerating the inference latency observed during execution.

Regarding the web engineering perspective, another promising technical direction involves the integration of the WebGPU API within the ONNX Runtime Web library. Liu et al. propose in their work an optimization technique for deep inference execution within browser environments [46]. Currently, the application relies exclusively on the device's CPU through WebAssembly to perform the model's matrix computations. Enabling the system to leverage the client's local graphics processing unit via WebGPU would substantially mitigate the current inference latency, resulting in a considerably more responsive experience for the end user.

Finally, in order to consider a more realistic practical deployment, it would be valuable to adapt the system to the constraints of smartphone cameras. As the model was trained exclusively on high-quality dermoscopic images provided by the ISIC Archive, its real-world performance may be affected by the comparatively lower image quality produced by mobile camera. A possible idea might be to incorporate domain-specific data during model training. About this, Pacheco et al. introduced a dataset composed exclusively of dermoscopic images captured with smartphones, known as PAD-UFES-20 [47]. Incorporating such images into the training pipeline would enhance the model's robustness and generalization capacity when dealing with lower-quality images, such as those from phone cameras.

Bibliography

[1] Ana F. Duarte, Bernardo Sousa-Pinto, Luís F. Azevedo, Ana M. Barros, Susana Puig, Josep Malvehy, Eckart Haneke, and Osvaldo Correia. Clinical abcde rule for early melanoma detection. *European Journal of Dermatology*, 31(6):771–778, Nov 2021.

[2] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[3] International Skin Imaging Collaboration. Isic archives. `https://www.isic-archive.com/`.

[4] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, 09–15 Jun 2019.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] ONNX contributors. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`.

[7] Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. Bringing the web up to speed with webassembly. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2017, page 185–200, New York, NY, USA, 2017. Association for Computing Machinery.

[8] ONNX Runtime contributors. Onnx runtime. `https://github.com/microsoft/onnxruntime`.

[9] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5(1):180161, Aug 2018.

[10] Teresa Mendonça, Pedro M. Ferreira, Jorge S. Marques, André R. S. Marcal, and Jorge Rozeira. Ph2 - a dermoscopic image database for research and benchmarking. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5437–5440, 2013.

[11] Konstantin Korotkov and Rafael Garcia. Computerized analysis of pigmented skin lesions: A review. *Artificial Intelligence in Medicine*, 56(2):69–90, 2012.

[12] Tim Lee, Vincent Ng, Richard Gallagher, Andrew Coldman, and David McLean. Dullrazor®: A software approach to hair removal from images. *Computers in Biology and Medicine*, 27(6):533–543, 1997.

[13] Howard Zhou, James M. Rehg, and Mei Chen. Exemplar-based segmentation of pigmented skin lesions from dermoscopy images. In *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 225–228, 2010.

[14] M. Emre Celebi, Hassan A. Kingravi, Bakhtiyar Uddin, Hitoshi Iyatomi, Y. Alp Aslandogan, William V. Stoecker, and Randy H. Moss. A methodological approach to the classification of dermoscopy images. *Computerized Medical Imaging and Graphics*, 31(6):362–373, 2007.

[15] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017.

[16] Hamed Tabrizchi, Sepideh Parvizpour, and Jafar Razmara. An improved vgg model for skin cancer detection. *Neural Processing Letters*, 55(4):3715–3732, Aug 2023.

[17] Arief Budhiman, Suyanto Suyanto, and Anditya Arifianto. Melanoma cancer classification using resnet with data augmentation. In *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 17–20, 2019.

[18] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[19] Sara Hosseinzadeh Kassani and Peyman Hosseinzadeh Kassani. A comparative study of deep learning architectures on melanoma detection. *Tissue and Cell*, 58:76–83, 2019.

[20] Vimalraj S Spelmen and R Porkodi. A review on handling imbalanced data. In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–11, 2018.

[21] Gehad Ismail Sayed, Mona M. Soliman, and Aboul Ella Hassanien. A novel melanoma prediction model for imbalanced data using optimized squeezenet by bald eagle search optimization. *Computers in Biology and Medicine*, 136:104712, 2021.

[22] Duyen N. T. Le, Hieu X. Le, Lua T. Ngo, and Hoan T. Ngo. Transfer learning with class-weighted and focal loss function for automatic skin cancer classification, 2020.

[23] Chih-Chi Chang, Yu-Zhen Li, Hui-Ching Wu, and Ming-Hseng Tseng. Melanoma detection using xgb classifier combined with feature extraction and k-means smote techniques. *Diagnostics*, 12(7), 2022.

[24] Yinhao Wu, Bin Chen, An Zeng, Dan Pan, Ruixuan Wang, and Shen Zhao. Skin cancer classification with deep learning: A systematic review. *Frontiers in Oncology*, Volume 12 - 2022, 2022.

[25] Fábio Perez, Cristina Vasconcelos, Sandra Avila, and Eduardo Valle. Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 303–311, Cham, 2018. Springer International Publishing.

[26] Balazs Harangi. Skin lesion classification with ensembles of deep convolutional neural networks. *Journal of Biomedical Informatics*, 86:25–32, 2018.

[27] Nils Gessert, Maximilian Nielsen, Mohsin Shaikh, René Werner, and Alexander Schlaefer. Skin lesion classification using ensembles of multi-resolution efficientnets with meta data. *MethodsX*, 7:100864, 2020.

[28] Andre G.C. Pacheco and Renato A. Krohling. The impact of patient clinical information on automated skin cancer detection. *Computers in Biology and Medicine*, 116:103545, 2020.

[29] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[30] Gan Cai, Yu Zhu, Yue Wu, Xiaoben Jiang, Jiongyao Ye, and Dawei Yang. A multimodal transformer to fuse images and metadata for skin disease classification. *The Visual Computer*, 2022.

[31] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Charles Nicholson, Nick Kreeger, Ping Yu, Shanqing Cai, Eric Nielsen, David Soegel, Stan Bileschi, Michael Terry, Ann Yuan, Kangyi Zhang, Sandeep Gupta, Sarah Sirajuddin, D Sculley, Rajat Monga, Greg Corrado, Fernanda Viegas, and Martin M Wattenberg. Tensorflow.js: Machine learning for the web and beyond. In *Proceedings of Machine Learning and Systems*, volume 1, pages 309–321, 2019.

[32] Hock-Ann Goh, Chin-Kuan Ho, and Fazly Salleh Abas. Front-end deep learning web apps development and deployment: a review. *Applied Intelligence*, 2023.

[33] Xu K Wang Z Maldonado F Sandler K Landman BA Huo Y. Dong C, Li TZ. Characterizing browser-based medical imaging ai with serverless edge computing:

towards addressing clinical data security constraints. In *Proceedings of SPIE Medical Imaging*, 2023.

[34] Jaisakthi S M, Mirunalini P, Chandrabose Aravindan, and Rajagopal Appavu. Classification of skin cancer from dermoscopic images using deep neural network architectures. *Multimedia Tools and Applications*, 82(10):15763–15778, Apr 2023.

[35] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636, 1998.

[36] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[37] Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, Mar 2019.

[38] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[40] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[41] Qishen Ha, Bo Liu, and Fuxu Liu. Identifying melanoma images using efficientnet ensemble: Winning solution to the siim-isic melanoma classification challenge, 2020.

[42] Anil Kumar Adepu, Subin Sahayam, Umarani Jayaraman, and Rashmika Arramraju. Melanoma classification from dermatoscopy images using knowledge distillation for highly imbalanced data. *Computers in Biology and Medicine*, 154:106571, 2023.

[43] Atheer Bassel, Amjed Basil Abdulkareem, Zaid Abdi Alkareem Alyasseri, Nor Samsiah Sani, and Husam Jasim Mohammed. Automatic malignant and benign skin cancer classification using a hybrid deep learning approach. *Diagnostics*, 12(10), 2022.

[44] Jose-Agustin Almaraz-Damian, Volodymyr Ponomaryov, Sergiy Sadovnychiy, and Heydy Castillejos-Fernandez. Melanoma and nevus skin lesion classification using handcraft and deep learning feature fusion via mutual information measures. *Entropy*, 22(4), 2020.

[45] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

[46] Fucheng Jia, Shiqi Jiang, Ting Cao, Wei Cui, Tianrui Xia, Xu Cao, Yuanchun Li, Qipeng Wang, Deyu Zhang, Ju Ren, Yunxin Liu, Lili Qiu, and Mao Yang. Empowering in-browser deep learning inference on edge through just-in-time kernel optimization. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, page 438–450. Association for Computing Machinery, 2024.

[47] Andre G.C. Pacheco, Gustavo R. Lima, Amanda S. Salomão, Breno Krohling, Igor P. Biral, Gabriel G. de Angelo, Fábio C.R. Alves Jr, José G.M. Esgario, Alana C. Simora, Pedro B.C. Castro, Felipe B. Rodrigues, Patricia H.L. Frasson, Renato A. Krohling, Helder Knidel, Maria C.S. Santos, Rachel B. do Espírito Santo, Telma L.S.G. Macedo, Tania R.P. Canuto, and Luíz F.S. de Barros. Pad-ufes-20: A skin lesion dataset composed of patient data and clinical images collected from smartphones. *Data in Brief*, 32:106221, 2020.

Appendix A

Resources and Demo

A.1 Data Availability

The source code for the contributions presented in this master's thesis is available in the following GitHub repository : `https://github.com/LoicHernaut/MelanomaDetector`

It contains:

- PYTORCH_MODEL - The multimodal's training code.
- ONNX_MODEL - The code for converting to the ONNX format, along with associated tests.
- app-melanoma - The source code for the melanoma detection web application.

A.2 Video Demonstration

A demo video of the 'MelanomaDetector' web app is available to observe the app in action: `https://youtube.com/shorts/RLNnalZSXEI`

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**