ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Computer Vision

SKINSCAN - RECOGNITION OF PIGMENTED SKIN LESIONS WITH VISION TRANSFORMERS AND BAYESIAN NETWORKS

CANDIDATE	SUPERVISOR
Marco Acerbis	Prof. Samuele Salti

Academic year 2022-2023

Session 3rd

dedicated to Alfa

Contents

List of Figures

List of Tables

Abstract

SkinScan aims to provide reliable, efficient, and cost-effective tools to assist physicians in identifying pigmented skin lesions. The application harnesses the capabilities of two modules, each handling distinct data types: an image classifier for analyzing lesion images and a Bayesian network for estimating the probability of developing a specific disease. The resulting system is deployed through a user-friendly Web App utilizing Kserve, making the trained models and algorithms accessible on a Kubernetes cluster.

Chapter 1

Introduction

In recent years, computer vision techniques based on Deep Learning (DL) have found extensive application in medical diagnosis. The ongoing challenge in modern medicine remains the effective detection of cancer, where identifying tumors in their early stages significantly enhances the chances of successful treatment and patient survival.
While skin cancers are some of the most common, melanoma diagnosis through visual examination exhibits an accuracy rate that varies from 65-70% without external support, to an increased 85% when technical assistance is available [1, 10]. However, clinical tests can pose difficulties, ranging from limited resources leading to prolonged waiting times and expensive examinations to the need for invasive procedures. Additionally, the accuracy rate for non-expert clinicians drops drastically between 20 to 40% [23], a clear indicator of the need for more accessible and reliable diagnostic tools.

This project aims to propose a solution to aid physicians during the diagnosis process with a fast, cheap and reliable tool. The system, named SkinScan and summarized in Figure 1.1, consists of two modules designed to operate on distinct types of input data. The first agent, outlined in Chapter 2, has been developed through the experimentation of various deep learning architectures

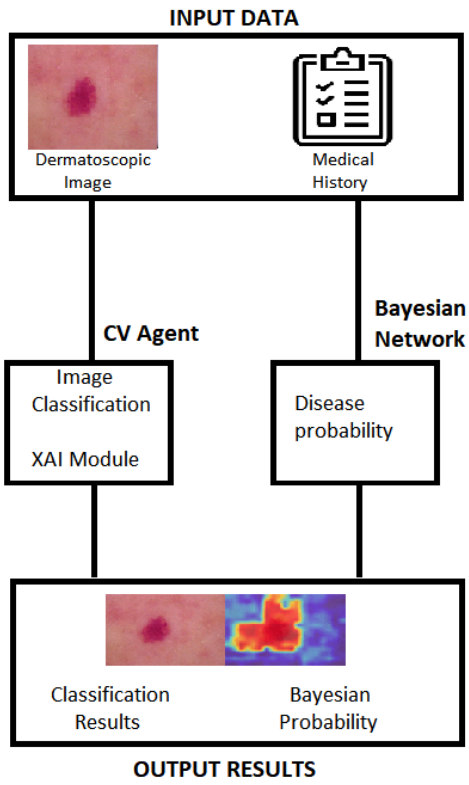

Figure 1.1: Summary of the proposed system.

for image classification. This process aims to establish a reliable tool for accurately identifying the seven most common pigmented skin lesions. Furthermore, it has been equipped with dedicated explainability algorithms to better highlight the relevant areas of the lesion under examination.

But patients are not only images of a lesion, they present a series of other relevant information, like the genetic exposure to cancer or showing particular signs, that can be summarized in the patient's medical history. These data are processed by a second agent; a Bayesian network designed to provide the probability of developing a given disease.

The final output includes both the results of image classification and the probabilities for each disease, presented with comparative images underling the most relevant areas of the lesion.

The final prototype has been developed to provide an easily accessible tool, that physicians can use on-the-go and that requires minimal external support.

For this purpose we realized, in collaboration with the IT company Intré, a web-app that deploys the SkinScan system, as illustrated in Chapter 4. In particular, we also wanted to create an instrument able to get more robust over time by learning from its mistakes and supported by larger datasets. This guided us into studying a solution that implements *Continuous Training* in our application, giving the system the ability to growth and adapt to new technologies and data.

Chapter 2

Image Classifier

The initial idea that started this project was a challenge proposed by the International Skin Imaging Collaboration (ISIC) to foster an active research on melanoma detection from dermatoscopic images of skin lesions and other non-invasive techniques. Their aim is to reduce melanoma-related deaths, accounting for the growing incidence of this type of cancer in the past years, by supplying high resolution images of different skin lesions to improve the diagnosis with the aid of AI.
In our research, we started by adapting and evaluating three different model architectures for image classification in order to find the optimal solution both in terms of accuracy and latency. After that, we developed two tools that help to explain the classification results by providing important insights about the salient areas of the image.

2.1 Models

While Convolutional Neural Networks (CNNs) proved to be an excellent tool to help physician with image classification [15], in recent years Vision Transformers (ViTs) have matched the performance of classical convolution approaches in almost every task. Given the aims of this project, we evaluated

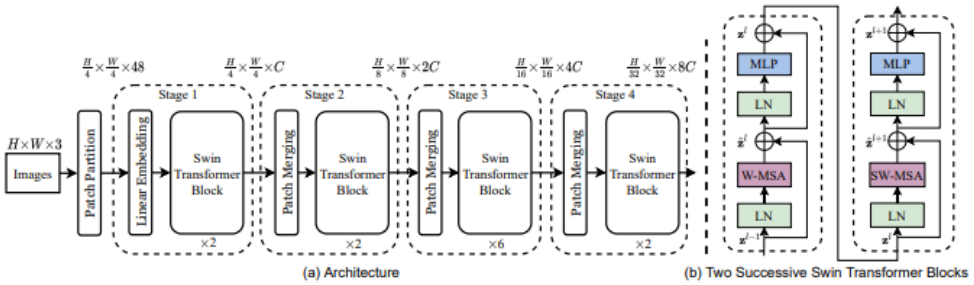

Figure 2.1: (a) Swin Transformer architecture, where each block makes use of (b) modules based on shifted windows rather than the standard multi-head self attention (MSA) module.

three different neural network architectures to find the optimal solution to implement in our application.

2.1.1 Swin-Transformer

Since their release in 2020, Vision Transformers (ViTs) [5] performances largely surpassed the ones of the state-of-the-art convolutional architectures in almost every task. But this dominance come at a cost. First, the original ViT architecture, adapted from Natural Language Processing (NLP), worked only with a fixed scale; a property in contrast with the majority of vision tasks the same object in multiple images can present different sizes. A second, but not less important, problem is working with high resolution images from the moment that the computational complexity of classical implementations of self-attention in ViTs is quadratic to image size. In order to overcome these limitations, Swin-Transformer [13] introduced a series of innovations. The proposed architecture builds hierarchical feature maps to handle objects at different scales, all while maintaining a linear computational complexity to image size by computing self-attention locally on non-overlapping windows that partition the image; as shown in Figure 2.2.

One of the main features introduced by Swin is the *shifted window* approach, depicted in Figure 2.3. In particular, the splitting over the image is

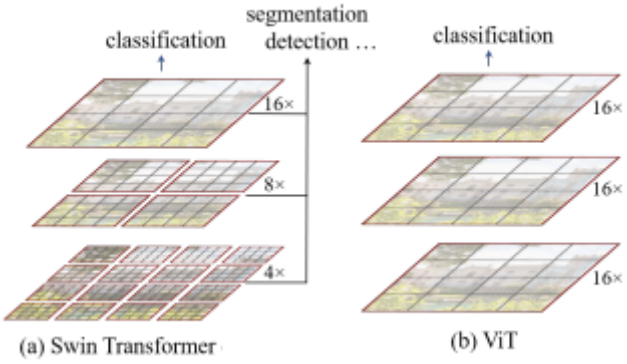

Figure 2.2: (a) Swin-Transformers creates hierarchical feature maps by merging image patches, differently from (b) the standard ViT approach that implements feature maps of a single resolution.

shifted between consecutive self-attention layers, thus connecting the windows of the two and enhancing the modeling power. This allows for local computation of self-attention on non-overlapping windows. Suppose now that in a window we have $M \times M$ patches and that the whole image is composed of $h \times w$ patches, then the computational complexities for global self-attention (MSA) and window-based self-attention (W-MSA) are reported in Equations 2.1 and 2.2 respectively.

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \tag{2.1}$$

$$\Omega(\text{W-MSA}) = 4whC^2 + 2M^2hwC. \tag{2.2}$$

These results show how the shifted window approach leads to linear computational complexity with respect to the input size, thus making Swin a suitable solution when working with high resolution images. The resulting architecture for Swin-T, the smallest version inside the Swin-Transformers family, is shown if Figure 2.1.

In our study, we worked with the latest version of this architecture, Swin Transformer v2 [12], that introduces a series of improvements aimed to scale up the model's capacity and resolution; as well as to reduce the needs for

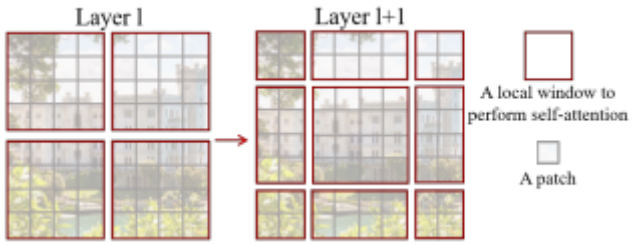

Figure 2.3: Illustration of the adopted shifted window approach used to perform self-attention in Swin-Transformer.

large amounts of labeled data. This last implementation, in particular, results quite important for our application, since we are bound to work with a limited amount of labeled data, compared to the huge datasets used for pre-training.

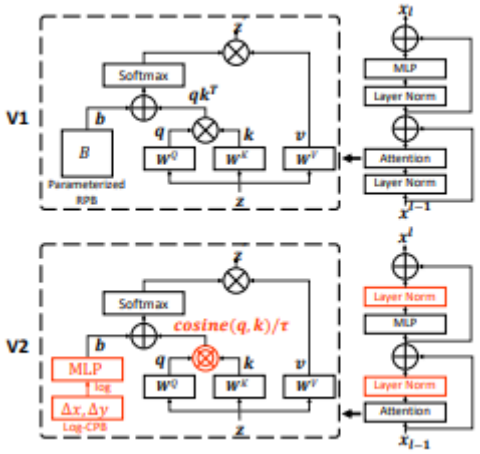

Figure 2.4: Comparison between Swin-Transformers v1 and v2 architectures.

In details, the overall architecture of Swin has not been changed significantly. Rather, a series of target improvements, highlighted in Figure 2.4, have been adopted:

- Instead of the *pre-norm* configuration, the new version implements a *res-post-norm* that moves *LayerNorm* from the beginning of each residual block to the back-end. This choice aims to solve a discrepancy of activation amplitudes across layers, thus making training more stable;
- A *scaled cosine attention* to replace previous *dot product attention*. This

modification has been introduced to address some instability issues arising during training, especially after adopting res-post-norm;

- The last improvements consist in adopting *log-spaced continuous* relative position bias, replacing the *parameterized* approach and allowing the model to be easier to transfer across window resolutions.

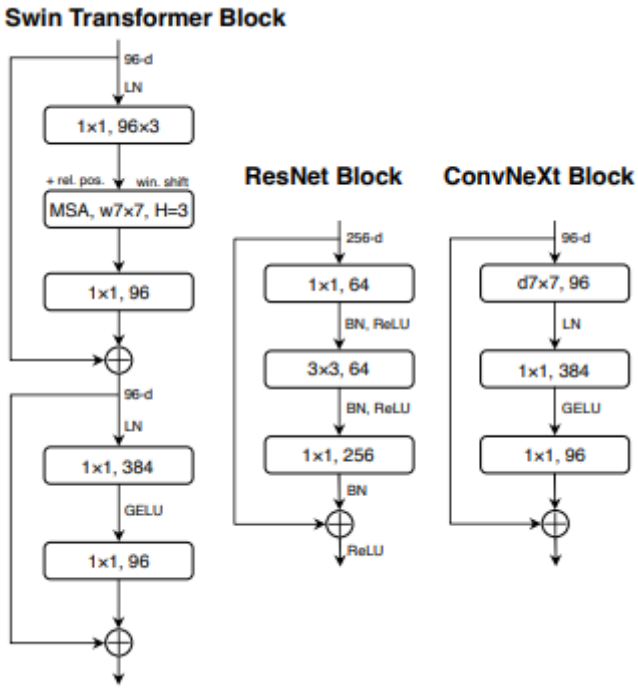

Figure 2.5: Blocks design comparison between Swin Transformer, ResNet and ConvNeXt.

2.1.2 ConvNeXt v2

Despite the large adoption of the transformer architecture in the computer vision field, during 2022 a new family of convolutional networks has been proposed. ConvNeXt [14] wants to refresh the ResNet [9] architecture with changes inspired by the dominant Transformers.

On an architectural point of view, the main innovation introduced by ConvNeXt is its block design, shown in Figure 2.5 compared with the previously discussed Swin-Transformer block and the standard ResNet one. ConvNeXt

implements an inverted bottleneck, but with a different layer order: by moving the depthwise convolution at the top, it allows for larger kernel size, passing from the standard 3×3 to bigger 7×7 convolutions. This adjustment makes the network's performance to increase, while the total FLOPs stay roughly the same.

A second important change to improve the performance is the removal of the stem layers in favor for a more Transformer-like "patchify" module. Normally, CNNs and transformers adopt stem layers to downsample (around 4×) the input image, i.e. reducing the spatial dimension while adding extra feature channels. On the other hand,Vision Transformers started to implement a more aggressive "patchify" strategy instead of the stem cell; which corresponds to a large kernel size (e.g. kernel size = 14 or 16) and non-overlapping convolutions. Like Swin, ConvNeXt adopts the same method, but with a 4×4 stride 4 convolutional layer that better fits the multi-stage architecture.

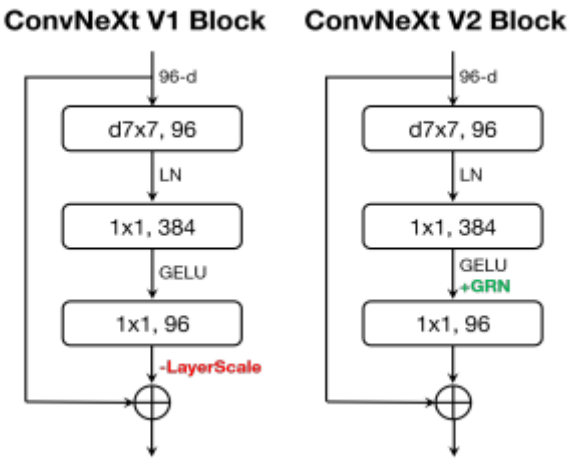

Figure 2.6: ConvNext v2 block compared with the original ConvNext v1 implementation.

Similarly to the approach used with Swin, we evaluated the performances of the more recent ConvNeXt v2 [25]. This updated version implements both *Masked Autoencoders* (MAE) and *Global Response Normalization* in the pretraining phase to increase the overall performance of the model with only minimal architecture changes. The idea of masked autoencoders derives from a

new way of exploring the design space using self-supervised pre-training. Differently from Vision Transformers applications, applying MAE to the ConvNeXt architecture is not direct: the encoder-decoder design of MAE is optimized to work with sequence processing, while CCNs apply dense sliding windows, and directly using masked inputs may lead to feature collapse at the MLP layer during training. To overcome these issues, the proposed architectures implements Fully Convolutional Masked Autoencoders (FCMAE) followed by Global Response Normalization (GRN) to avoid feature collapse. In details, given an input $X \in \mathbb{R}^{H \times W \times C}$, GRN is divided into three steps:

1. **Global feature aggregation**, where the spatial feature map X is aggregated into a vector gx with a function $\mathcal{G}(\cdot)$:
   $\mathcal{G}(X) = X \in \mathbb{R}^{H \times W \times C} \rightarrow gx \in \mathbb{R}^{C}$;

2. **Feature normalization** through a response normalization function $\mathcal{N}(\cdot)$:
   $\mathcal{N}(||X_i||) = ||X_i|| \in \mathbb{R} \rightarrow \frac{||X_i||}{\sum_{j=1...C} ||X_j||} \in \mathbb{R}$
   where $||X_i||$ is the L2-norm of the i-th channel;

3. **Feature calibration** using the computed feature normalization scores:
   $X_i = X_i * \mathcal{N}(\mathcal{G}(X)_i) \in \mathbb{R}^{H \times W}$.

To facilitate optimization, two learnable parameters, γ and β, are added along with a residual connection between input and output of the GRN layer. The resulting GRN block can be summarized as:

$$X_i = \gamma * X_i * \mathcal{N}(\mathcal{G}(X)_i) + \beta + X_i \tag{2.3}$$

This implementation makes the GRN layer to initially perform an identity function, and then gradually adapting during training. In Figure 2.6 we can see that the general block structure has remained almost invariant; GRN in particular makes the LayerScale unnecessary, explaining its removal.

2.1.3 SwiftFormer

As we have seen in section 2.1.1, the usage of the Transformer architecture, i.e. the self-attention mechanism, rapidly adapted from the original NLP applications to vision related tasks. Nevertheless, CNNs still maintained a dominant role in many real-time implementations, especially on mobile devices where the resources are limited. This happens because transformers are limited by the quadratic complexity of self-attention. SwiftFormer [22] aims to introduce an efficient additive attention mechanism to replace the expensive matrix multiplication and introduce a linear complexity with respect to the input size.

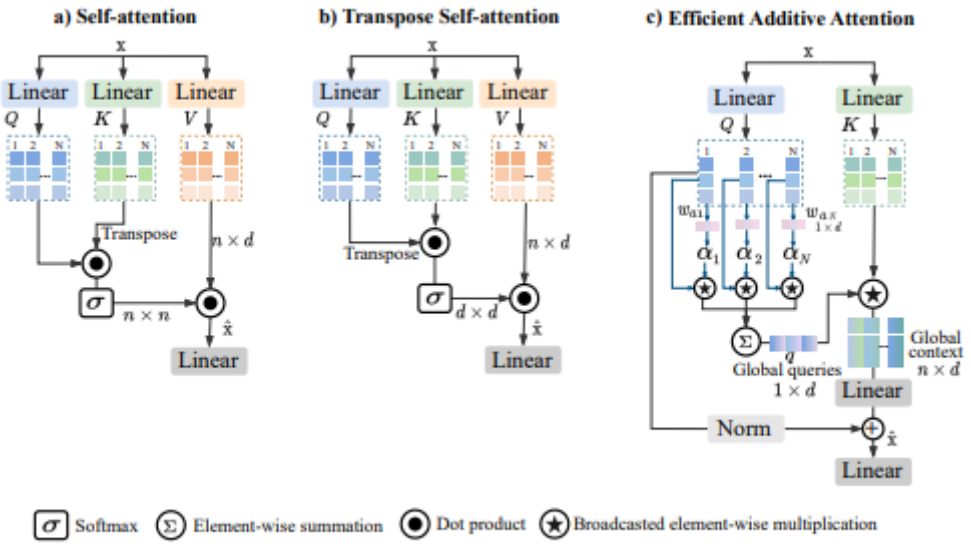

Figure 2.7: Different type of self-attention modules.

Transformers rely on self-attention that models the relationships between the input tokens, patches for ViTs, in the following way: the input $x \in \mathbb{R}^{n \times d}$, a set of n tokens with $d-$dimensional embedding vector, is projected to query (**Q**), key (**K**) and value (**V**) using three matrices, namely $\mathbf{W_Q}$, $\mathbf{W_K}$, $\mathbf{W_V}$. Note that in Multi-Head Self Attention (MHSA) each attention layer is equipped with h heads so that the model can learn different views of the input. The

standard self-attention, shown in Figure 2.7 (a), can be summarized as follow:

$$\hat{x} = Softmax\left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d}}\right) \cdot \mathbf{V} \tag{2.4}$$

From Equation 2.4, we can derive that the complexity, as expected, is equal to $O(n^2 \cdot d)$, where n is the number of tokens and d the hidden dimension. A tentative solution to this problem would be using transpose attention, Figure 2.7 (b), defined as:

$$\hat{x} = \mathbf{V} \cdot Softmax\left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d}}\right) \tag{2.5}$$

The resulting complexity of Equation 2.5 is $O(d^2 \cdot n)$; linear respect to the number of tokens n, so to the input size, but that scales quadratically with the feature dimension d.

The proposed efficient additive attention, schematized in Figure 2.7 (c), removes the key-value interactions to focus only on effectively encoding the query-key interactions by adopting broadcasted element-wise multiplication. Differently from the previous self-attention modules, the query matrix $\mathbf{Q}$ is now multiplied by a learnable parameter vector $\mathbf{w}_a \in \mathbb{R}^{d \times d}$ to obtain the attention weights and the consequential global attention query vector $\alpha \in \mathbb{R}^n$ defined as:

$$\alpha = \mathbf{Q} \cdot \frac{\mathbf{w}_a}{\sqrt{d}} \tag{2.6}$$

At this point the query matrix is pooled accordingly to the learned attention weights, generating a single global query vector $\mathbf{q} \in \mathbb{R}^d$:

$$\mathbf{q} = \sum_{i=1}^{n} \alpha_i * \mathbf{Q}_i \tag{2.7}$$

Finally, the interactions between $\mathbf{q}$ and the key matrix $\mathbf{K}$ are encoded using the element-wise product to get global context and a linear transformation layer is added to learn the hidden token representations. The final output can be described as:

$$\hat{x} = \hat{\mathbf{Q}} + \mathbf{T}\left(\mathbf{K} * \mathbf{q}\right) \tag{2.8}$$

where $\hat{\mathbf{Q}}$ is the normalized query matrix and $\mathbf{T}$ is the linear transformation. While this method generates similar matrices as MHSA, it is less expensive to compute and has linear complexity with respect to the input size.

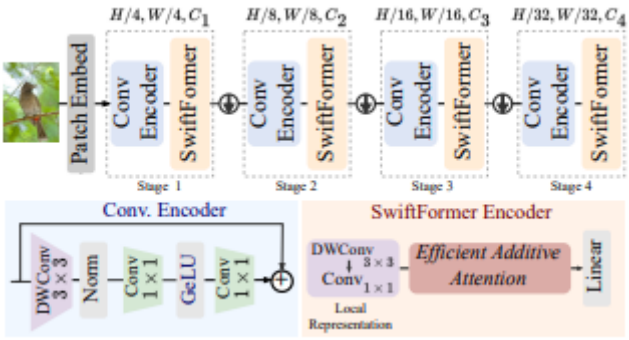

Figure 2.8: SwiftFormer architecture with details about the conv. encoder (bottom left) and the SwiftFormer Encoder (bottom right)

SwiftFormer, schematized in Figure 2.8, is also an example of hybrid architecture: to overcome the fixed scale of ViTs it introduces, in its *Patch Embedding* layer, 3×3 convolutions to extract spatial features at four different scales. After that, each stage deploys a combination of **Conv. Encoder**, designed to learn local representations through 3×3 depth-wise convolutions, and **SwiftFormer Encoder**, that learns the enriched local-global representations.

2.2 Dataset

For our evaluation experiments, we used 450 × 600 px dermatoscopic images available through the HAM10000 dataset [24]. This collection gathers 10015 images of the seven most common pigment skin lesions: Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec, 1099), basal cell carcinoma (bcc, 514), benign keratosis-like lesions (bkl, 1099), dermatofibroma

(df, 115), melanoma (mel, 1113), melanocytic nevi (nv, 6705) and vascular lesions (vasc, 142).
As we can see from the number of images per class, the dataset is highly unbalanced with around 67% of the images belonging to the melanocytic nevi class. At the same time , two classes (dermatofibroma and vascular lesions) covers only around 1% of the available data. A proposed solution to this problem is to expand the available dataset with synthetic data [15] by adopting generative techniques to provide images for the more rare classes. While this method can provide realistic images of skin lesions, they lack the effective correlation with an actual disease and there are no tools to evaluate their accuracy in mimic actual skin damage. Given these consideration, we decided to tackle the unbalance of the HAM10000 dataset by implementing Focal Loss [11], as detailed in the next section.

Starting from the whole dataset of 10015 images, we divided it into three subsets in the following way:

- **Train/Test Split** with a ratio of 70/30 of the whole dataset to generate the first two subsets: Train (7010 images) and Test (3005 images);
- **Training/Validation Split** with a ratio of 90/10 of the Train set to obtain the Training (6309 images) and Validation Sets (701 images).

2.3 Model Selection

In our project we need to implement a reliable image classifier to help physicians in early detection of skin cancer. So as to establish which of the proposed models performs the best in our task, we designed a series of experiments to evaluate each architecture performances.

2.3.1 Focal Loss

As we have seen in section 2.2, the HAM10000 dataset is imbalanced. In order to limit the disproportions between the number of images for each class, we trained our models twice: at first each model has been trained with the standard Cross-Entropy Loss (CE Loss) used in multi-labels classification problems; then we repeated the training/validation cycle with Focal Loss [11].
Originally developed to optimize object detection and discriminate between background and foreground classes, Focal Loss (FL) aims to overcome the class imbalance problem and to help in the detection of the more rare classes.

$$CELoss = -\sum_{i=1}^{n} y_i log(p_i) \tag{2.9}$$

Equation 2.9 defines the standard categorical Cross-Entropy (CE) Loss, where y_i is the ground truth and p_i is the probability for the i^{th} class given n classes. A typical problem with CE when working with imbalanced datasets is that the dominant class examples overwhelm the loss function, thus the gradient descend, and the model tends to neglect the other classes. With Focal Loss, we down-weight easy examples, i.e. the ones belonging to the dominant class, and focus the training on the others. The implemented Focal Loss is defined as:

$$FL(y, \hat{\mathbf{p}}) = -(1 - \hat{p}_y)^{\gamma} log(\hat{p}_y) \tag{2.10}$$

where:

- $y \in \{0, ..., n-1\}$ is an integer class label (n is the number of classes);
- $\hat{\mathbf{p}} = (\hat{p}_0, ..., \hat{p}_{n-1}) \in [0, 1]^n$ is a vector with estimated probability distribution over the K classes;
- γ is the *focusing parameter* that specify how much higher-confidence correct predictions contribute to the overall loss.

As we can see from Equation 2.10, FL is equivalent to CE Loss when $\gamma = 0$. At the same time higher the value of γ increase the effect of the modulating factor $(1 - \hat{p}_y)$.

(CE Loss)	Swin-Transformer v2	ConvNeXt v2	SwiftFormer
Train Accuracy	99.3%	98.9%	90.5%
Val accuracy	92.4%	92.1%	85.6%
Train Loss	0.02	0.03	0.05
Val Loss	0.55	0.47	0.12
(Focal Loss)	Swin-Transformer v2	ConvNeXt v2	SwiftFormer
Train Accuracy	99.3%	98.9%	90.0%
Val accuracy	93.7%	92.6%	84.9%
Train Loss	0.004	0.01	0.05
Val Loss	0.08	0.06	0.13

Table 2.1: (Top) Training and validation results for each model with Cross-Entropy Loss. (Bottom) Training and validation results for each model using Focal Loss.

2.3.2 Training and Validation

The first step in our evaluation consists in two training cycles of 50 epochs for each model; the first using the standard Cross-Entropy Loss, defined in Equation 2.9, while the second has been conducted implementing the previously discussed Focal Loss, reported in Equation 2.10.

The detailed configurations for each model are reported in Appendix A.

Swin-Transformer v2

From Figure 2.9, we can clearly identify the presence of overfitting when Swin was trained with the standard CE-Loss; an issue that has partially been solved when we switched in favor of the Focal Loss. From Table 2.1, we notice another significant effect of FL: the validation accuracy rose from 92.4% to 93.7%. This suggests that we are successfully classifying more instances from the less frequent classes without excessively penalizing the dominant one.

About the computational time, each complete train/validation cycle took between 4 to 5 hours to be completed.

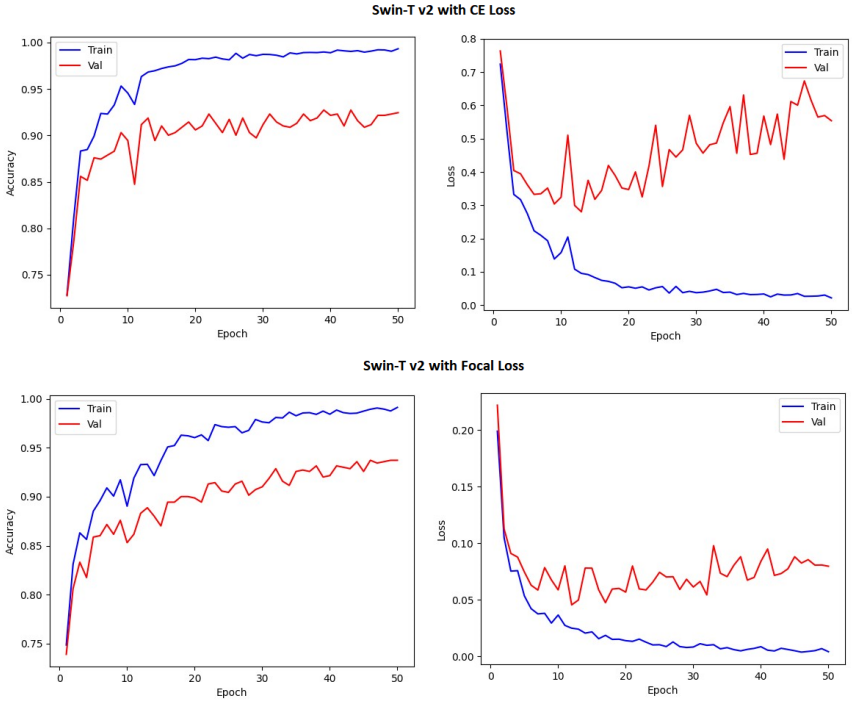

Figure 2.9: Accuracy and loss for the training cycles of Swin-Transformer v2.

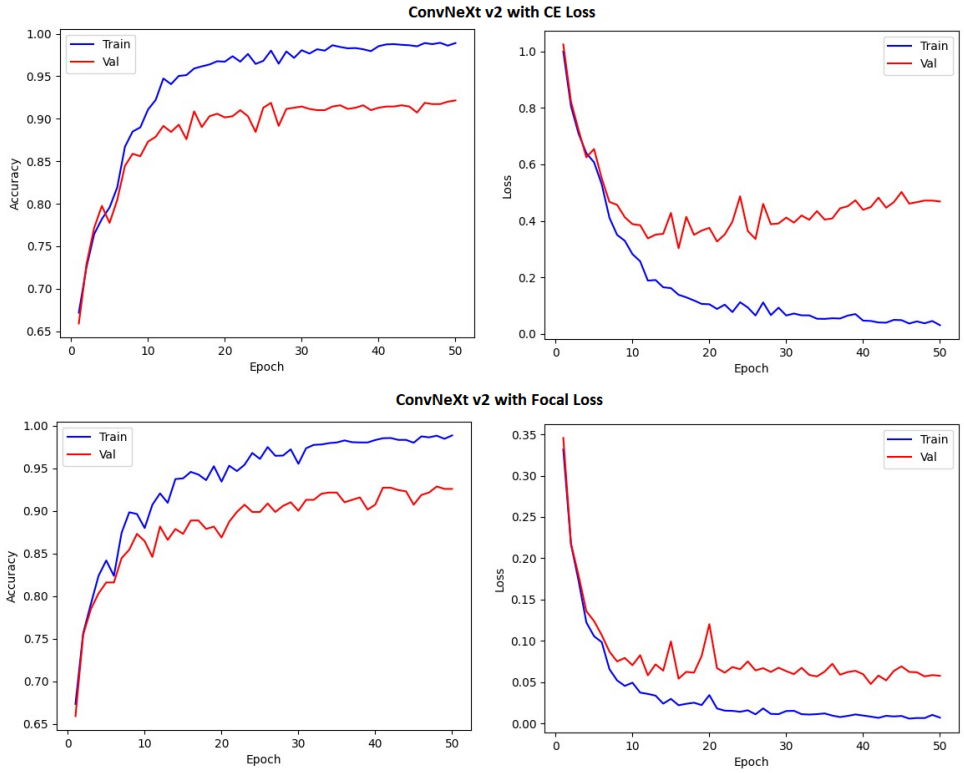

Figure 2.10: Accuracy and loss for the training cycles of ConvNext v2.

ConvNeXt v2

Figure 2.10 illustrates the impact resulting from the introduction of Focal Loss over Cross-Entropy in the training of ConvNeXt. As confirmed by the results reported in Table 2.1, this change not only considerably reduced the overfitting, but also increased the the validation accuracy from 92.1% to 92.6%.
On the down side, ConvNeXt needed between 10 to 11 hours and a more powerful GPU to complete each training/validation cycle.

SwiftFormer

The results in Table 2.1 shows that the implementation of the Focal Loss did not positively impacted the training of SwiftFormer, with a general decreased in the accuracy performances and almost unchanged losses. Furthermore, The graphs in Figure 2.11 depict the absence of relevant overfitting, even when the model was being trained with Cross-Entropy Loss. This results can be explained by analyzing how the Focal Loss works. Whit FL, we are trying

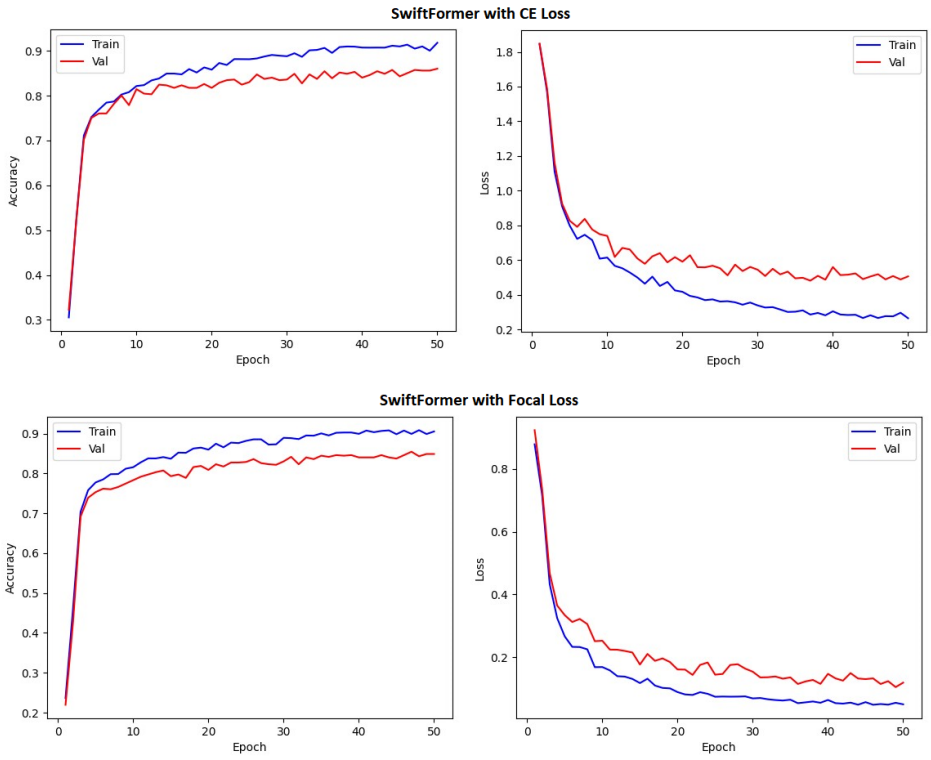

Figure 2.11: Accuracy and loss for the training cycles of SwiftFormer.

to focus the training on the examples from the more rare classes in order to improve their correct classification rate. On the other hand, we are partially 'neglecting' the examples belonging to the more frequent classes, in particular the ones of the dominant class 'nv'. So, the end result sees a modest increase in accuracy, deriving from the correct classification of few rare examples, that is overcome by an accuracy loss due to more misclassified examples in the dominant class. This supposition has been later confirmed from the test results, discussed in the next section.

On a positive note, SwiftFormer turned out to be the fastest and less computationally demanding model, completing each cycle in roughly 45 minutes.

2.3.3 Test results

Each trained model has been tested on 3005 images and to evaluate the results we adopted two metrics: the general accuracy in correctly classifying the skin

	Swin-Transformer v2	ConvNeXt v2	SwiftFormer
Test Accuracy	92.0%	91.9%	84.7%
AKIEC	71.9% (64/89)	73.0% (65/89)	59.5% (53/89)
BCC	91.8% (135/147)	90.5% (133/147)	76.2% (112/147)
BKL	82.1% (274/335)	81.8% (275/335)	68.6% (230/335)
DF	89.1% (41/46)	78.3% (36/46)	36.9% (17/46)
MEL	72.2% (244/338)	77.2% (261/338)	49.4% (167/338)
NV	97.8% (1962/2005)	97.2% (1950/2005)	96.0% (1925/2005)
VASC	100% (45/45)	95.6% (43/45)	93.3% (42/45)
	Swin-Transformer v2	ConvNeXt v2	SwiftFormer
Test Accuracy	91.8%	91.8%	85.0%
AKIEC	77.5% (69/89)	76.4% (68/89)	51.7% (46/89)
BCC	89.1% (131/147)	88.4% (130/147)	80.3% (118/147)
BKL	82.2% (275/335)	81.5% (273/335)	67.2% (225/335)
DF	84.8% (39/46)	80.4% (37/46)	54.3% (25/46)
MEL	72.8% (246/338)	75.1% (254/338)	52.4% (177/338)
NV	97.4% (1953/2005)	97.3% (1951/2005)	95.8% (1921/2005)
VASC	100% (45/45)	100% (45/45)	95.5% (43/45)

Table 2.2: (Top)Test results for each model with Cross-Entropy Loss. (Bottom) Test results for each model using Focal Loss.

lesions, and the per-class accuracy. The second metric not only allows us to assess the model's ability in identifying rare examples but also enables us to examine the types of misclassifications and errors that occurred.

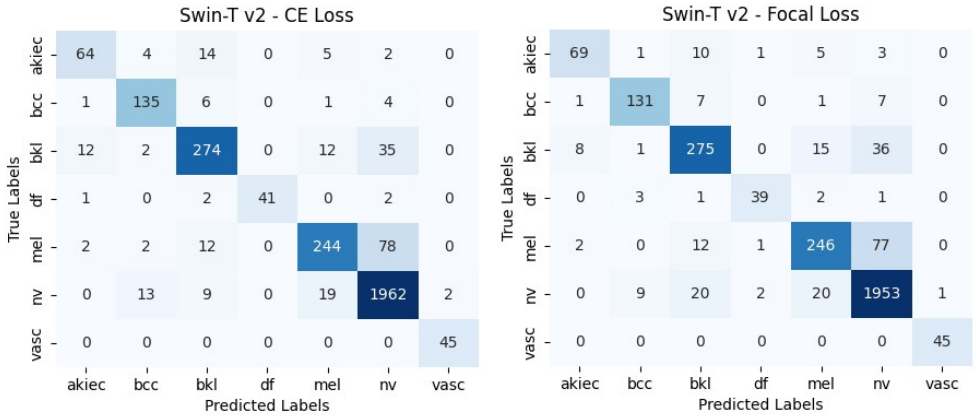

Figure 2.12: Confusion Matrices for Swin-Transformer v2.

Swin-Transformer v2

From the results in Table 2.2, we observe that Swin is, in general, the best performing model between the ones we studied, with an overall test accuracy

of 92.0% (CE Loss) and 91.8% (FL). Just as observed in the training of SwiftFormer, we can notice a marginal decrease of 0.2% in test accuracy between the model implementing CE Loss and the one utilizing Focal Loss. The confusion matrices presented in Figure 2.12 offer deeper insights. As anticipated in the previous section, the adoption of Focal Loss led to improved performance in classifying rare examples. For instance, the 'akiec' rate increased from 64 out of 89 (71.9%) to 69 identified images (77.5%), indicating a general improvement with 5 more correctly classified lesions. However, this comes at a cost, as the accuracy in the dominant class 'nv' decreased from 1962 to 1953. Here, nine images were misclassified by the second model, resulting in a net performance decrease of 4 more lesions not correctly identified. Interestingly, the adoption of Focal Loss affected negatively the classification of samples belonging to the rare 'df' class, whose recognition rate slightly dropped from 41 to 39.

A further examination reveals which are the most commonly mistaken classes. The first pattern we can notice is that both models commits the same errors when classifying actinic keratoses (akiec) and benign keratosis-like lesions (bkl) due to their intrinsic similarities: both diseases are a form of keratosis, a condition characterized by the formation of 'scales' on the skin surface, and produce similar lesions. An analog mistake is the incorrect classification of melanomas as melanocytic nevi, since both disease may produce deformed moles or similar pigmentation as shown in Figure 2.13.

After this analysis, we can conclude that the accuracy loss deriving from the implementation of the Focal Loss is negligible, and that the second model is the optimal one from the moment it was able to better focus on 2 out of 3 rare classes with a contained loss in the general accuracy.

ConvNeXt v2

Table 2.2 indicates that ConvNeXt performs similarly to Swin, with a test accuracy around 91.9%. From the confusion matrices in Figure 2.14 we observe that the models also share error patterns, like many 'bkl' samples classified

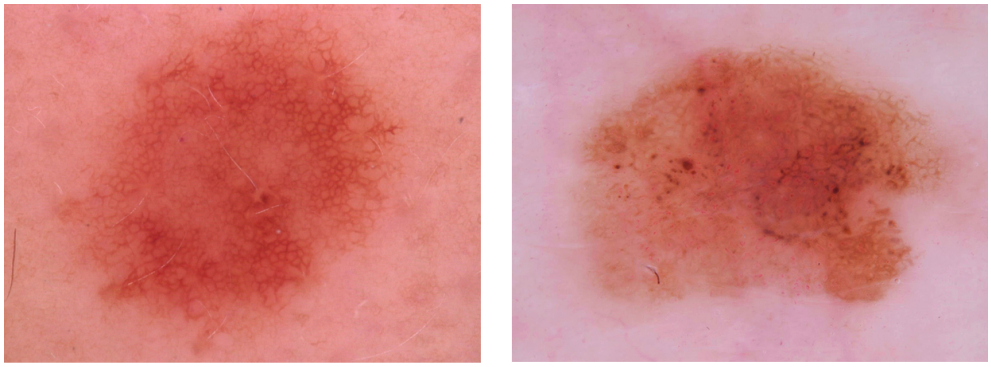

Figure 2.13: Comparison between a melanocytic nevi (left) and a misclassified melanoma (right).

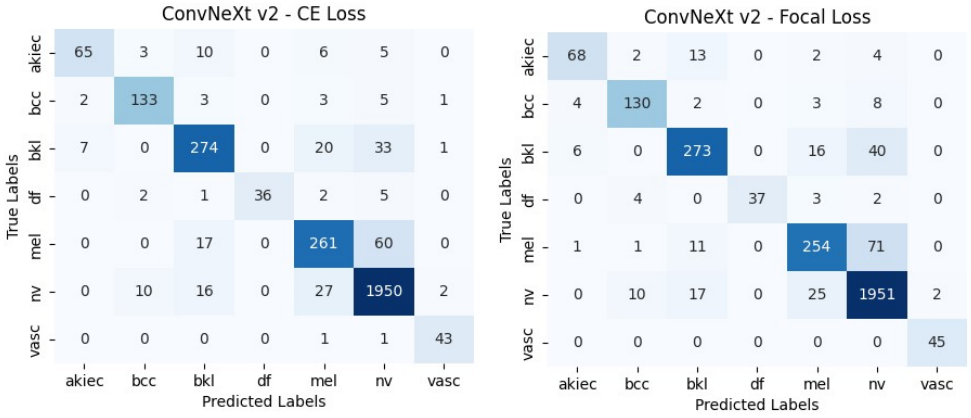

Figure 2.14: Confusion Matrices for ConvNeXt v2.

as melanocityc nevi. This misclassification, that arise independently from which loss function we utilize, may derive from the alike pigmentation that the growths develop in their early stages, but it is not a dangerous mistake from the moment that moles and bkl are non-cancerous conditions. On the contrary, the misidentification of melanomas as melanocytic nevi, increased from 60 to 71 instances when Focal Loss has been used, is more risky from the moment that a cancerous formation is being confused with a natural skin condition.

Overall, ConvNeXt v2 displays strong performance in recognizing pigmented

skin lesions. However, there might be a need to focus on reducing the misclassification of melanomas if the model is to be integrated into the final application.

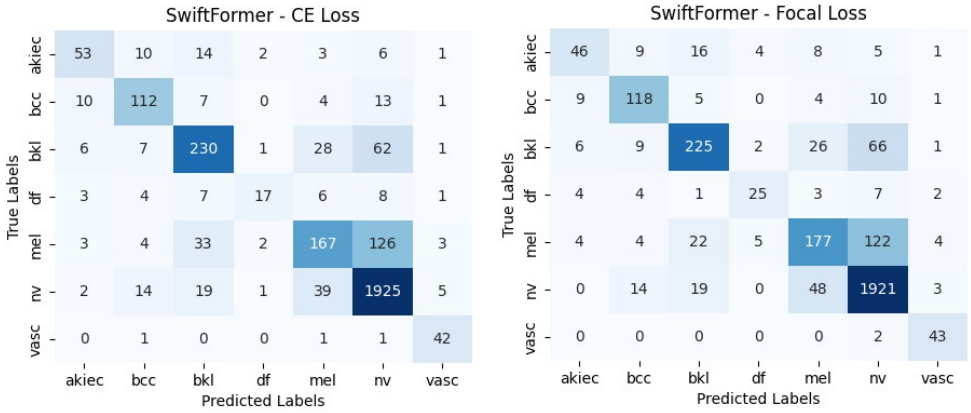

Figure 2.15: Confusion Matrices for SwiftFormer.

SwiftFormer

Last column of Table 2.2 reports the rusults obtained by SwiftFormer. As expected from the previously discussed training, the model has the worst performances among the three studied here, with a test accuracy that ranges from 84.7% with CE Loss to 85.0% when FL is used. Despite the small increase in the overall accuracy, the introduction of the Focal Loss only partially helped in improving the model performance. From the confusion matrices in Figure 2.15, we can see an increase in detection rate of dermatofibromas, from 17 to 25, and a more modest one in the recognition of vascular lesion with an extra image being correctly classified. At the same time, the third rare class, 'akiec', saw a reduction in the accuracy rate, moving from 53 correctly identified samples to only 46 out of 89. Furthermore, both models presents increased error rates in every category when compared to Swin or ConvNeXt. In particular, almost half of the instances of the melanoma class had been wrongly classified as melanocytic nevi; making this model unsuitable, at least, as a tool for the recognition of melanoma.

In conclusion, taking into account that the inference times of the three proposed models are similar, we have chosen to implement Swin-Transformer trained with Focal Loss in our application. In the final evaluation comparing Swin and ConvNeXt results, the superior performance in identifying melanoma over melanocytic nevi has been recognized as crucial.

2.4 Explainable AI (XAI)

A big obstacle that AI applications have to face, especially in the medical field, is that they are usually seen as a black-box that produce results following its own logic, rather then the correct diagnostic reasoning. Furthermore, many physicians remain sceptical [15] and prefer to not follow or even do the opposite of what an algorithm suggests. At the same time, some of these biases find some reflection in reality: for example the image of a lesion may include external elements that can alter the outcome of the analysis. To overcome these issues, we decided to introduce two tools dedicated to the explanation of the classification provided by our agent.

In particular, we opted for two different solutions that aims to present different aspects of the analysis: Grad-CAM and Deep Feature Factorization (DFF).

2.4.1 Grad-CAM

Introduced in 2017, *Gradient-weighted Class Activation Mapping* (Grad-CAM) [21] is used to create localization map that underlines the most relevant pixels used by a CNN to classify an image with respect to a given class, as shown in Figure 2.16. CAM [26] is a localization technique for identifying discriminative regions, i.e. the portions of the image that are more relevant, to compute image classification. This initial approach to CNNs' explainability is limited by the need of CNNs that do not contain any fully-connected layer, limiting the

usage of this method only on models that implement Global Average Pooling. Given this limitation, Grad-CAM introduces a new way of combining feature maps using the gradient signal, allowing applications of this approach to be applicable to any CNN-based architecture.

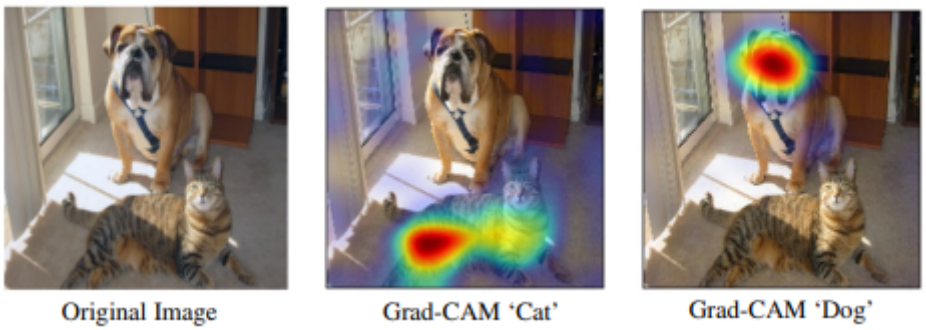

Figure 2.16: GradCAM used to shows the relevant pixels for two different classifications: 'cat' (middle Figure) or 'dog' (right Figure)

We know that convolutional layers do retain spatial information, thus we can expect the last layers to have the best compromise between high-level semantics and detailed spatial information. In particular, the neurons in these layers are tuned to identify semantic class-specific information in the image, the so-called object parts. Grad-CAM uses the gradient information to assign importance values to each neuron for a particular decision of interest. In order to generate the localization map $L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$ of width u and height v for any given class c, first the gradient of the score for class c y^c (before the softmax) is computed with respect to feature maps activations A^k of a convolutional layers, namely $\frac{\partial y^c}{\partial A^k}$. These gradients flowing back are then global-average-pooled over the spatial dimensions u and v to obtain the *neuron importance weights* α^c_k, as in Equation 2.11:

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{2.11}$$

Each weight α^c_k represents a *partial linearization* of the deep network downstream and capture the 'importance' of feature map k for a target class c.

The final localization map $L^c_{Grad-CAM}$ ensues from a weighted combination of forward activation maps followed by a ReLU, as in Equation 2.12:

$$L^c_{Grad-CAM} = ReLU\left(\sum_k \alpha^c_k A^k\right) \tag{2.12}$$

Note that ReLU is applied to the linear combination of maps in order to extract only the *positive* influence towards the class of interest c, or rather those pixels whose intensity should be *increased* if we want to increase the class score y^c. Without this operation, the produced heatmap may include *negative* pixels that, more likely, belong to objects of different classes and thus presenting a worse localization.

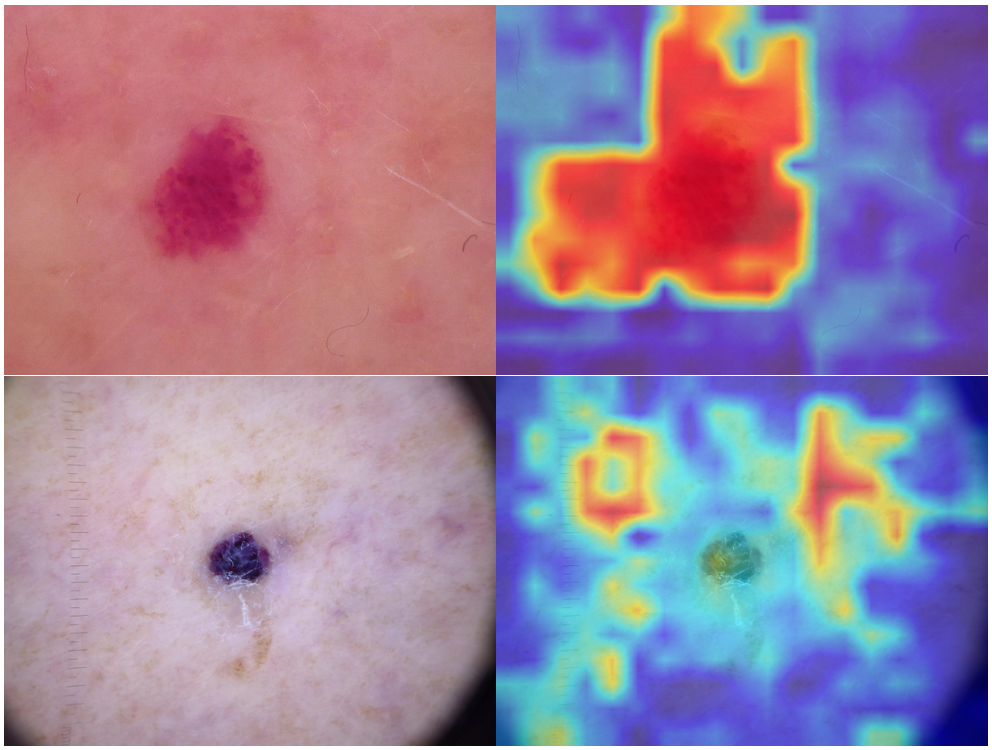

Figure 2.17: (top) Example of Grad-CAM applied to the image of a correctly classified vascular lesion. (bottom) Grad-CAM can help us identify wrongly classified examples by showing that no relevant pixels have been used to identify the lesion.

From Equation 2.11, we can also see that y^c needs to be any differentiable activation, thus making Grad-CAM adaptable to other architectures different

from CNNs; and in particular it can be modified to work with vision transformers. The seconds present both forward and backward pass that are needed to compute the gradients, and the attention weights provide a form of relevance information that can used to find the salient regions of the image for a particular prediction. This flexibility of Grad-CAM allowed us to implement it on Swin, by selecting the last layer before the final classification (softmax), in order to get an explanation of the whole network process [6, 21]. The examples shown in Figure 2.17 illustrate how Grad-CAM can help in validating the results obtained by the image classifier (top image) or discard them when no relevant part of the image or external elements, like measuring scales or hair, have been used for the classification (bottom image).

2.4.2 Deep Feature Factorization

Being able to distinguish between skin lesions of different origin can be sometimes difficult, especially when there are not clear indicators that can help distinguish between two or more classes. Deep Feature Factorization (DFF) [3] is a localization technique that introduces a way of finding similar semantic concepts inside an image aiming to answer the question *'what does the classifier see in this picture?'*.

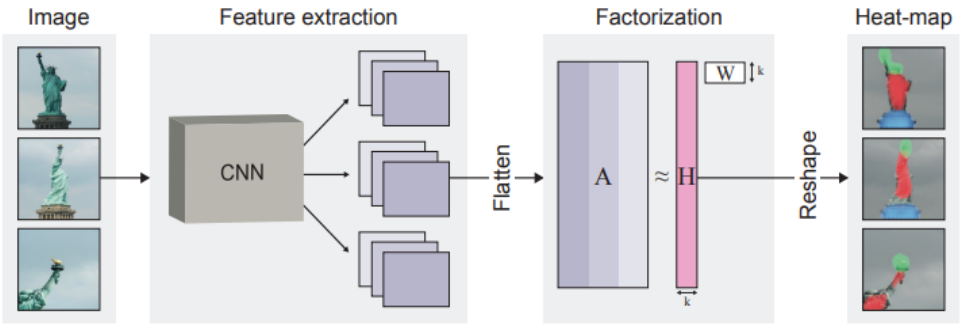

Figure 2.18: Graphical representation of Deep Feature Factorization.

The main idea behind DFF, schematized in Figure 2.18, is to utilize Non-Negative Matrix Factorization (NMF) to find patterns among the activations

of a pre-trained neural network model.

With factorization methods, we search for an approximation of the form

$$\begin{aligned} A \approx \hat{A} = HW \\ \text{s.t. } A, \hat{A} \in \mathbb{R}^{n \times m}, H \in \mathbb{R}^{n \times k}, W \in \mathbb{R}^{k \times m} \end{aligned} \tag{2.13}$$

where $\hat{A}$ is a low-rank matrix of rank k.

A data point, i.e. a row of A, is represented as a weighted combinations of the factors which form the rows of W. In particular, when A is non-negative we can perform NMF defined as:

$$\begin{aligned} &NMF(A, k) = argmin_{\hat{A}_k} \|A - \hat{A}_k\|_F^2, \\ &\text{subject to } \hat{A}_k = HW, \forall i, j H_{ij}, W_{ij} \geqslant 0, \\ &\text{and where } \|\cdot\|_F \text{ is the Frobenius norm} \end{aligned} \tag{2.14}$$

This way we capture the structure in A while forcing combinations of factors to be additive results in factors that lend themselves to interpretation.

Both CNNs and Transformers see an input image I as tensor of dimension $h_I \times w_I \times c_I$, where h_I and w_I are the height and width respectively and c_I is the number of color channels. In particular, it can been seen as a c_I-dimensional feature representation of a spatial position. After ℓ layers, we obtain the feature map A_I^ℓ of dimension $h_\ell \times w_\ell \times c_\ell$ which has a spatial representation similar to the original image I: while the first two dimensions represent a spatial grid for a given patch of pixels in I, the last forms a c_ℓ-dimensional representation of the patch. In particular, the deeper the ℓ layer is the more abstract and semantically meaningful features vectors are, hence the name *Deep Features*. Each feature map is made of multiple patches and we can view them as points in the same c_ℓ-dimensional space (*feature space*), thus allowing us to study the presence of patterns among the vectors representing similar patches.

Given the activation tensor of an image I at layer ℓ:

$$A_I^\ell \in \mathbb{R}_+^{h \times w \times c} \tag{2.15}$$

we first flatten tensor A into a matrix of dimension $(h \cdot w) \times c$, this way the spatial arrangement is lost and the rows of A can be permuted without affecting the result of factorization.

$$A_I^\ell \in \mathbb{R}_+^{(h \cdot w) \times c} \tag{2.16}$$

Now, by applying NMF to A we obtain the matrices H and W defined in Equation 2.13:

- **W Matrix**

  Each row W_j ($1 \leqslant j \leqslant k$) forms a c-dimensional vector in the feature space. Since NMF is performing clustering, we view a factor W_j as a centroid of an activation cluster, which we show corresponds to coherent object.

- **H Matrix**

  This matrix presents as many rows as matrix A, each corresponding to a spatial position in the image. Instead, each column H_j ($1 \leqslant j \leqslant k$) can be reshaped into n heatmaps of dimension $h \times w$, which highlight regions in the image I that correspond to the factor W_j. Note that those heatmaps have the same spatial dimension as the layer which produced the activation. Thus, to match the input image size, they are upsampled using bilinear interpolation.

Predicted Class	Score
VASC (6)	0.515
NV (5)	0.478
MEL (4)	0.004
BKL (2)	0.001
BCC (1)	9e-4
DF (3)	5.3e-5
AKIEC (0)	1.4e-9

Table 2.3: Example of classification outputs.

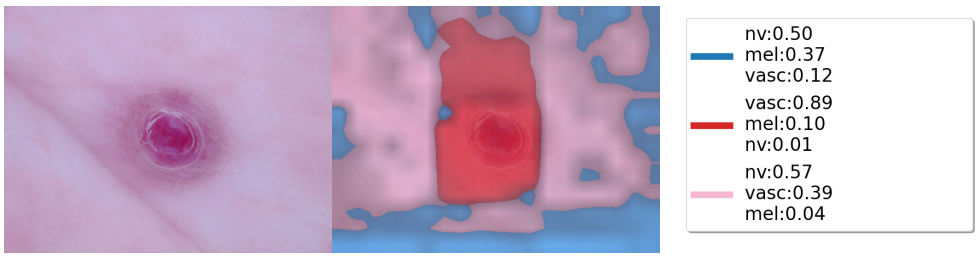

Figure 2.19: DFF example on a dermatoscopic image that get a doubtful classification.

DFF turns out useful especially when the classification is doubtful, i.e. two o more classes get a very similar score. For example, Figure 2.19 shows an image that gets the correct label ('*vasc*') with a score of only 51.5%, while the second top scoring label ('*nv*') receives a score of 47.8%, as reported in Table 2.3. In this scenario, a tool like DFF can be crucial in understanding what and where the classifier 'sees' objects, thus leading the physicians towards a correct diagnosis.

Chapter 3

Bayesian Networks

Bayesian Networks (BN) are an example of probabilistic graphical models in the form of directed acyclic graphs (DAG) in which each node represents a random variable and the edges indicate a probabilistic relationship between two nodes. Furthermore, each node has an associated Conditional Probability Table (CPT), also referred to Conditional Probability Distribution (CPD), that specifies the probability distribution of the node given its parents, i.e. the nodes directly connected to it.
Bayesian Networks are also a popular tool in the medical field and are employed for different tasks that range from disease diagnosis [4] to treatment planning. In our application, a bayesian network provides an efficient and reliable approach to handle all the non-visual information about the patient's medical history; for that the proposed model has been developed under the supervision of a qualified dermatologist, Dr. Marzia Baldi, that acted as domain expert by providing important insights about the dependencies between risk factors, diseases and signs.

3.1 Proposed Model

The proposed model has been developed by following three steps:

1. For each of the studied diseases, we researched their *Incidence Rate* to

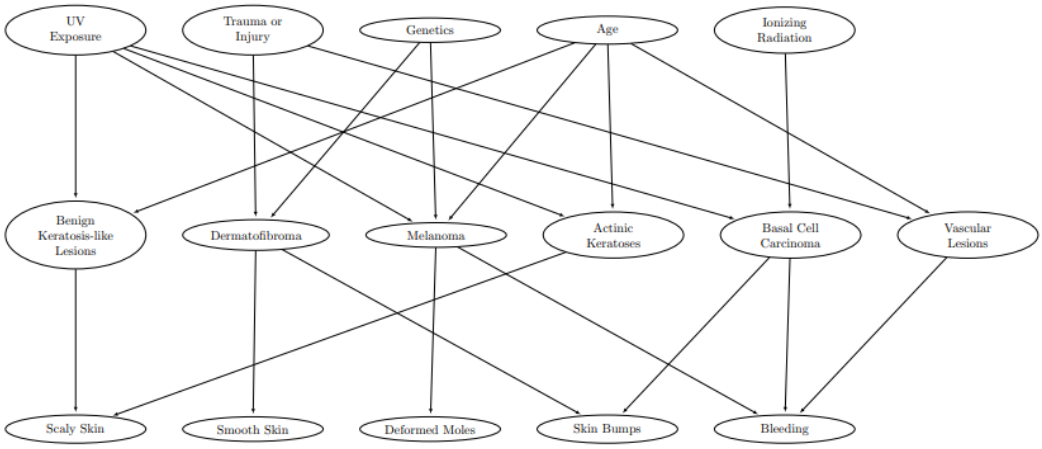

Figure 3.1: Illustration of the proposed bayesian network.

use as the base probability of developing the illness. It is important to note that melanocytic nevi are not considered a disease, given that moles are natural skin lesions commonly developing on the skin surface;

2. We identified and selected five of the most common *Risk Factors* that increase the probability of developing a given disease;

3. Finally, we collected five possible *Signs* that are not correlated to the visual information. Notably, we do not call them symptoms, a term that indicate a subjective patient-dependant indication. Instead, we prefer the definition of sign as an observable manifestation of a certain condition. Also, we are working with oncological illnesses that are usually asymptomatic.

The final model is depicted in Figure 3.1 from which we can notice the presence of the three macro levels described in previous steps: risk factors, diseases and signs. All the variables in our model are boolean variables, except for 'Age' which consists of three groups (under 14 yo, 14-65 yo, over 65 yo).

UV	T	T	T	T	T	T	F	F	F	F	F	F
GEN	T	T	T	F	F	F	T	T	T	F	F	F
AGE	0	1	2	0	1	2	0	1	2	0	1	2
MEL_T	0.00015	0.0003	0.000225	0.00012	0.00012	0.00024	0.000075	0.00015	0.0001125	0.00006	0.00012	0.00009
MEL_F	0.99985	0.9997	0.999775	0.99988	0.99988	0.99976	0.999925	0.99985	0.9998875	0.99994	0.99988	0.99991

Table 3.1: CPD describing the probability distribution for variable 'Melanoma'.

Table 3.1 shows an example of CPD Table describing the probability distribution of variable 'Melanoma'. The associated parent nodes are 'UV Exposure', 'Genetic predisposition' and 'Age'. In Italy, the average incidence rate of Melanoma ranges between 5 and 7 cases every 100'000 inhabitants [20]. Notably, excessive exposure to sunlight and other UV rays sources doubles the probability of developing the disease. Additionally, family history and genetic factors can elevate the probability by approximately 25% (x1.25). Furthermore, the incidence of this cancer increases notably between the ages of 25 to 50 years. To account for this age-related trend, we have defined multipliers of x1, x2, and x1.5 corresponding to the three age groups mentioned earlier. The complete list of the defined CPD tables is provided in Appendix B.

3.2 Inference

After defining our Bayesian Network, we explored various inference methods to query the model. A relevant aspect for our application is to ensure both reliability and fast processing of output probabilities when presented with a set of evidence.

3.2.1 Exact Inference

From Figure 3.1, we observe that the graph (DAG) defying our model is not a polytree from the moment that there are nodes connected with multiple paths, like the nodes 'UV Exposure' and 'Bleeding' that are connected by two different paths. This aspect may create some difficulties when we try to compute

Algorithm 1 Variable Elimination

inputs: Query variable X, evidence **e**, Bayesian network bn

factors ← [ ]
for each *var* **in** Order **do**
  factors ← [MAKE-FACTOR(*var*,**e**)| *factors*]
  if *var* is a hidden variable **then** *factors* ← SUM-OUT(*var*,*factors*)
return NORMALIZE(POINTWISE-PRODUCT(*factors*))

exact inference from the moment that the time complexity can become exponential with respect to the number of evidence variables. However, despite dealing with a total of 24 connections between nodes, we aim to perform exact inference to generate preliminary results. These results will serve as reference values for evaluating the effectiveness of approximate methods.
To compute exact inference we implemented *Variable Elimination*, a simple yet powerful algorithm for exact inference.

$$\mathbf{P}(X|) = \alpha\mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y}) \tag{3.1}$$

Given a query on a Bayesian Network in the form described in Equation 3.1, where X is a variable, **e** a set of evidence variables, **y** are the hidden variables and α is a normalization constant, we can develop each term of $\mathbf{P}(X, \mathbf{e}, \mathbf{y})$ as product of conditional probabilities from the network. Consequentially, we can resolve the query by computing sums of products. The detailed procedure applied by Variable Elimination is described in Algorithm 1.

3.2.2 Approximate Inference

While developing our SkinScan system, an important focus was made on continuous training. The final application does include the possibility to growth in the future, not only with better models for image classification, but also with stronger and more complex Bayesian Network that covers other diseases. Form these consideration arise the necessity of implementing a feasible algorithm for computing inference on our model.

Algorithm 2 Rejection Sampling

inputs: Query variable X, evidence **e**, Bayesian network bn, total number of samples to be generated N
local variables: **N**, a vector of counts for each value of X (starting at 0)

for j = 1 to N **do**
  x ← PRIOR-SAMPLE(bn)
  if **x** is consistent with **e** **then**
    N[x] ← **N**[x] + 1 where x is the value of X in **x**
return NORMALIZE(**N**)

Approximate inference methods, also called *Monte Carlo* algorithms, are based on different sampling techniques for creating instances that reflect the probability distribution specified in the Bayesian Network. Afterwards, the queried posterior probabilities are extracted from the generated population. We evaluate two approximate strategies:

Rejection Sampling

As outlined in Algorithm 2, rejection sampling starts by generating instances following the prior distribution specified in the network. Then, it proceeds to eliminate all those that do not match the evidence **e**. Finally, it estimates the posterior probability $P(X = x|\mathbf{e})$ by counting how often $X = x$ occurs in the remaining samples.

The main issue with this approach is that it creates the samples without using the information contained in the evidence. This aspect shows the main limitation of the approach: with more complex models and larger number of evidence variables, the number of samples compatible with the evidence drops exponentially making the rejection sampling inefficient.

Likelihood Weighting

Differently from the previous rejection approach, this sampling technique generates events that are already compatible with the provided evidence (importance sampling), as summarized in Algorithm 3. To account for the probability distribution specified by the network, it assigns a weight to each generated

Algorithm 3 Likelihood-Weighting

function LIKELIHOOD-WEIGHTING(X,**e**,bn,N) **returns** an estimate of **P**(X| **e**)
inputs: Query variable X, evidence **e**, Bayesian network bn, total number of samples to be generated N
local variables: **W**, a vector of weighted counts for each value of X (starting at 0)

for j = 1 to N **do**
  x,$w \leftarrow$ WEIGHTED-SAMPLE(bn,**e**)
  W[x] $\leftarrow$ **W**[x] + w where x is the value of X in **x**
return NORMALIZE(**W**)

function0: WEIGHTED-SAMPLE(bn, e) returns an event and a weight
$w \leftarrow 1$; **x** $\leftarrow$ an event with n elements initialized from **e**
for each variable $X_i in X_i ... X_n$ **do**
  if X_i is an event variable with value x_i in **e then**
    $w \leftarrow w \times P(X_i = x_i | parents(X_i))$
  else x[i] $\leftarrow$ a random sample from **P**($X_i | parents(X_i)$)
return x, w

sample that is proportional to its probability. In the end, the posterior probability is estimated as:

$$P(X = x|\mathbf{e}) = \alpha \sum_{y} N_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) \tag{3.2}$$

where N_{WS} is a generated sample with its relative wight w, and α is a normalization constant.

On a negative note, this technique performance tends to degrade as the number of evidence increase. This is because most samples will end with very small weights, making their contribution in the final estimation becomes negligible.

3.2.3 Evaluation

In order to evaluate the described inference methods, we run a series of experiments by changing the queried disease and the number of provided evidence

variables. Note that we limited the inference time to 5 minutes, after which we considered the outcome as a failure (NA) even if an outcome has been later produced.

Disease	Evidence variables	Exact Inference	Sample size	Rejection Sampling	Time (RS)	Likelihood Weighting	Time (LW)
AKIEC (T)	10	16.74%	10000	17.02%	24 s	4.32%	1s
AKIEC (T)	5	11.44%	10000	11.41%	3 s	4.44%	1s
MEL (T)	10	4.82%	10000	NA	>300 s	0%	1s
MEL (T)	4	0.3%	10000	NA	>300 s	0%	1s
BCC (T)	8	0.16%	10000	0.17%	7 s	4.32%	1s
BCC (T)	4	0.29%	10000	0.42%	3 s	0.49%	1s
DF (T)	10	0.05%	10000	NA	>300 s	0.07%	1s
DF (T)	5	0.07%	10000	NA	>300 s	0.07%	1s

Table 3.2: Inference methods evaluation results.

The results are reported in Table 3.2. As anticipated, Likelihood Weighting emerged as the top-performing algorithm for approximate inference, while Rejection Sampling failed to produce valid answers in four cases. Additionally, instances where the estimated posterior probability of Likelihood Weighting significantly deviated from the one obtained through exact inference generally involved a substantial number of provided evidence variables. This contrasts with real-world applications where physicians typically work with a limited set of signs and knowledge about the exposition to risk factors. In conclusion, we implemented Likelihood Weighting in our application.

Chapter 4

Web App

4.1 Web App Architecture

SkinScan is designed to serve as a readily accessible tool, aiding physicians and dermatologists in the diagnostic process. Given the frequent travel of many medical practitioners between clinics, coupled with their common access to laptops and digital dermatoscopes (or high-resolution cameras), the optimal deployment method for our system is via a web app. Our serving infrastructure consists of three modules that communicate:

4.1.0.1 Inference Service

First, we need a way to make our trained model and the relative XAI tools available as services through APIs. *Kserve* is a tool which enables serverless inferencing of machine learning (ML) models on Kubernets; compatible with the most common ML frameworks, like TensorFlow or Pytorch. In depth, Kserve allows to deploy our trained model, using a dedicated *Docker* container, and makes it available by exposing API endpoints inside a Kubernets cluster. This allows applications or services to send inference requests to the deployed model at different levels exploting the Kserve structure depicted in

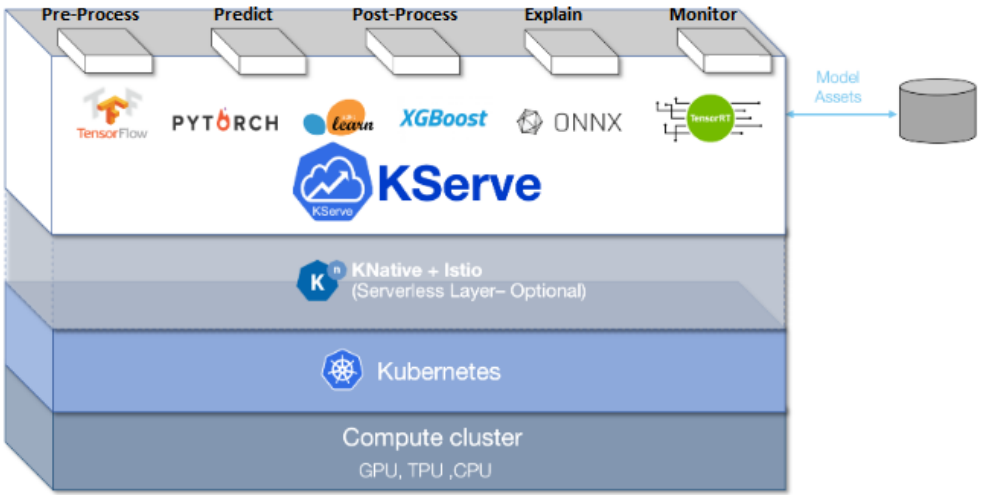

Figure 4.1: Illustration of the different levels provided by Kserve.

Figure 4.1. Furthermore, Kserve gives the possibility to set-up different function for prediction, by allowing the user to choose between a standard predict module or the creation of a custom one, and for the explainer. In our case, we implemented a custom predictor that performs inference on the uploaded image, while the explainer consists of two functions which apply Grad-CAM and Deep Feature Factorization respectively.

The serverless structure is enabled by the *Knative platform*, which provides aids to build, deploy and manage serverless workloads, and by *Istio*, a service mesh to control microservices through traffic management. In a similar way, Kserve offers a dedicated module that generates training pipelines inside the cluster, enabling an efficient way to implement continuous training in our application.

A crucial aspect of Kserve is its scalability. Thanks to the Kubernets environment, that manages to scale resources based on demand, our system can scale horizontally permitting multiple instances of our model that run in parallel. Also, Kubernets offers auto-scaling so that the system can create more instances (replicas) when a certain resource, like CPU or memory, exceeds a given threshold in order to meet the demand; and at the same time scale down the resources when demand is low.

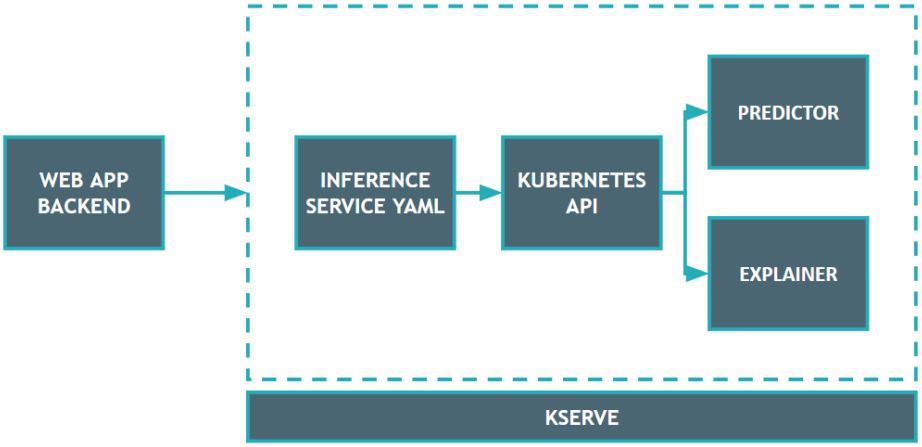

Figure 4.2: Illustration showing the communication between the backend and the inference module.

4.1.0.2 Backend

Our backend (BE), that we denominated *Derma*, provides a series of functions that form the backbone of our system:

- **Bayesian Network**

  The bayesian network, described in Chapter 3, is stored inside the BE module. Inference is triggered through a call when patient's medical history is provided;

- **Data Management**

  BE provides functions that collects the input data, images and questionnaire answers, and make them available to other services. For example, once an image is uploaded it is stored and pre-processed by BE, while the generated tensor is send to the inference module to obtain the prediction, as shown in Figure 4.2. Similarly, the bayesian network receives the medical history and proceeds to estimate the probabilities. All these outputs are once again collected by the BE which stores and provides them to the frontend that displays the results to the user;

- **Request Management**

  BE also controls the requests received by the application and handles them in order to optimize the resource usage and process multiple request in parallel;

- **Continuous Training**

  Once the models' outputs are collected, BE is also responsible to store them for future validation and comparison, but also for the creation of new training dataset used for continuous training. BE is also responsible of triggering the pipeline orchestrator, developed using Argo Workflows, which starts and manages new training tasks on our Kubernets cluster.

4.1.0.3 Frontend

As the final interface, the one which directly communicate with the user, the frontend (FE) consists of a UI page created with *React-JS*, a common library for this kind of applications. Figure 4.3 shows the final implementation: the user can upload an image of a skin lesion and answer a brief questionnaire about the patient's medical history (top image), from here the input data is collected by the backend services to be processed and then the final results are showed in the updated page (bottom image).

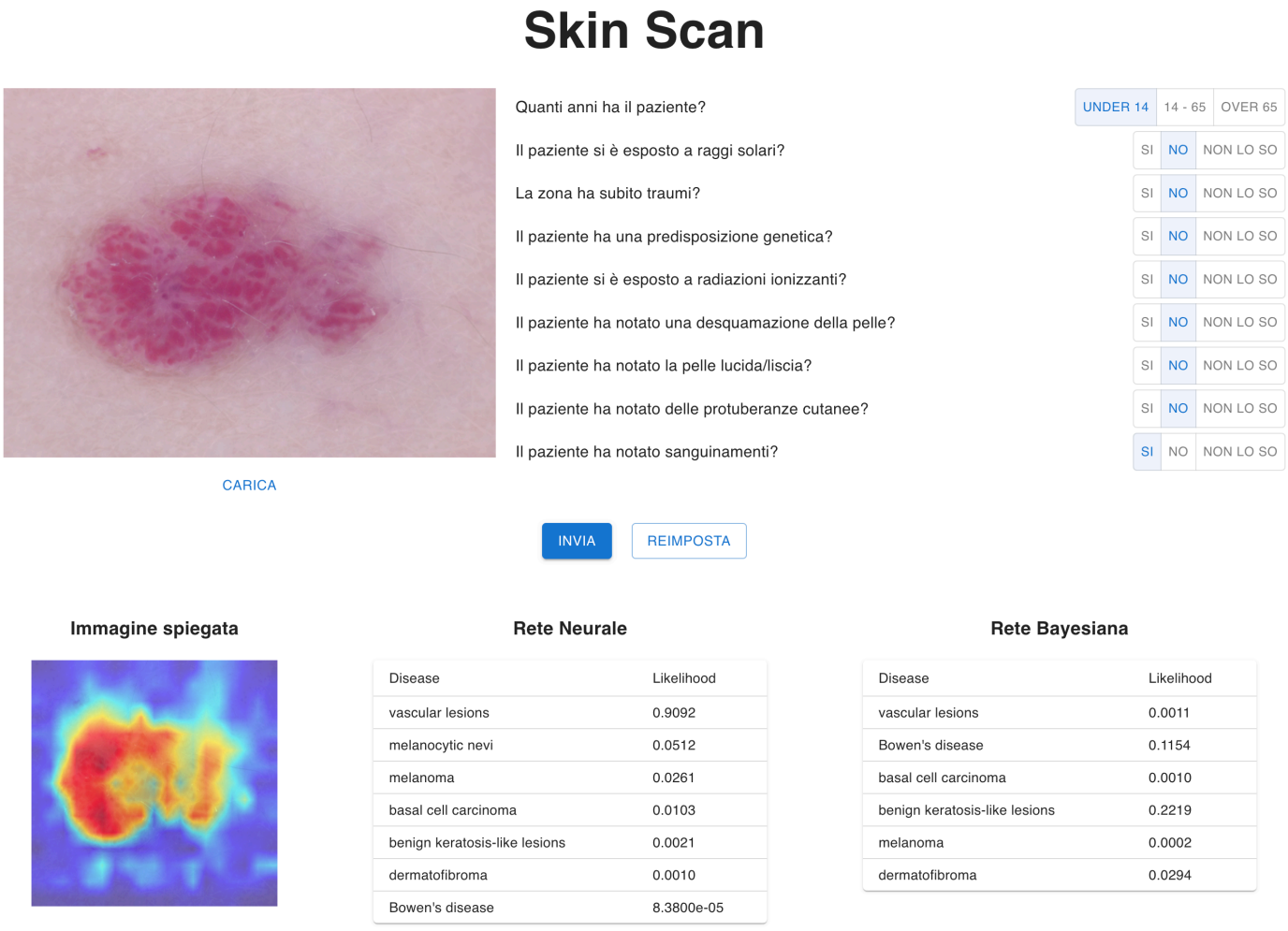

Figure 4.3: (top) UI to input the patient data (image of the lesion and medical history). (bottom) The final results, combining classification outputs, XAI heatmaps and bayesian probabilities, is shown to the user.

Chapter 5

Conclusion

In this thesis, we focused on realizing an efficient and reliable tool to help practitioners and dermatologists in recognizing pigmented skin lesions. As result, we developed the SkinScan system, that implements both Vision Transformers and Bayesian Networks, and deployed it through a Web App.

In Chapter 2 we studied how to apply the most recent Vision Transformers to our classification problem. In particular, we had to deal with the identification of rare classes, usually overwhelmed by examples of more frequent and normal lesions such as moles. To this end, we introduced the Focal Loss in our study to mitigate the problem. We then fine-tuned and tested three different architectures, namely Swin-Transformer v2, ConvNeXt v2 and SwiftFormer, to find the optimal solution for our task. After a careful considerations of the final results in Table 2.2 and of the errors, Swin has been selected as the model to implement in our application.

Furthermore, we studied two different approaches to Explainable AI. The first was Grad-CAM, a localization technique that allows the user to visualize which pixels were most relevant in the classification process. This tool gives physicians a way to better understand the reasoning followed by the Transformer and to discard the result when it gets negatively impacted by the

presence of external or non-relevant elements. To a similar end, we also studied a second technique, Deep Feature Factorization, that gives us the ability to better understand which lesions the model sees inside a picture.

In Chapter 3 we presented a statistical approach to medical diagnosis with Bayesian Networks. These models are based on probabilities and provide us with a practical tool for estimating the existence of malignant lesions. The proposed model, developed under the supervision of dermatologist Marzia Baldi, has been studied starting from the incidence rate of the studied diseases. After that, we recognised the principal risk factors, at the origin of the increased probability of developing a skin lesions, and the most common signs produced by dermatological illnesses.
Given the crucial aspect of providing a system that can grow over time, we also searched the optimal way to carry out inference on Bayesian Networks. We found the desired answer in Likelihood Weighting, that also better suits the realistic scenarios in which physicians work with limited evidence knowledge.

Finally, in Chapter 4 we showed how to deploy the system through a Web App and, thanks to the help provided by Intré, we presented our prototype of a final product. Here, we implemented a solution based on Kserve, a tool that makes our trained models available on a Kubernets cluster where they become accessible to applications and services.

Appendix A

Models Configurations

All the pre-trained models of the studied architectures are available on *Huggin Face* through their *Transformers* library:

- Swin-Transformer v2: microsoft/swinv2-base-patch4-window12-192-22k;
- ConvNeXt v2: facebook/convnextv2-base-1k-224;
- SwiftFormer: MBZUAI/swiftformer-xs

Training parameters and set-up:

- Batch size: 16;
- number of epochs: 50;
- Focusing parameter γ: 5;
- Learning rate=5e-5;
- Gradient accumulation steps: 4;
- Warmup ratio: 0.1 (with linear scheduler);
- GPUs: Nvidia Geforce RTX 3070 (Swin and SwiftFormer), Nvidia V100 (ConvNeXt)

Full code available at: https://github.com/Ace95/SkinScan

Appendix B

CPD Tables

Diseases CPDs:

- Actinic keratoses [19, 2]
  - Average incidence rate: 0.044;
  - UV exposure: 2;
  - Age: 1,1.3,1.8;

UV	T	T	T	F	F	F
AGE	0	1	2	0	1	2
AKIEC_T	0.088	0.1144	0.1584	0.044	0.0572	0.0792
AKIEC_F	0.912	0.8856	0.8416	0.956	0.9428	0.9208

Table B.1: CPD describing the probability distribution for variable 'Actinic keratoses'.

- Basal Cell Carcinoma [16]
  - Average incidence rate: 0.00146;
  - UV exposure: 2;
  - Ionizing radiation: 2.3;

UV RAD	T T	T F	F T	F F
BCC_T	0.006716	0.00292	0.003358	0.00146
BCC_F	0.993284	0.99708	0.996642	0.99854

Table B.2: CPD describing the probability distribution for variable 'Basal Cell Carcinoma'.

- Bening Keratoses-like Lesions [8]
  - Average incidence rate: 0.3;
  - UV exposure: 2;
  - Age: 1,1.3,1.5;

UV AGE	T 0	T 1	T 2	F 0	F 1	F 2
BKL_T	0.6	0.78	0.9	0.3	0.39	0.45
BKL_F	0.4	0.22	0.1	0.7	0.61	0.55

Table B.3: CPD describing the probability distribution for variable 'Bening Keratosis-like Lesions'.

- Dermatofibroma [18, 7]
  - Average incidence rate: 0.03;
  - Trauma or Injury: 2;
  - Genetics: 1.2;

TRA GEN	T T	T F	F T	F F
DF_T	0.072	0.06	0.036	0.03
DF_F	0.072	0.06	0.036	0.03

Table B.4: CPD describing the probability distribution for variable 'Dermatofibroma'.

- Vascular Lesions [17]

  - Average incidence rate: 0.0005;

  - Trauma or Injury: 2;

  - Age: 0.7,1.1,1.5;

TRA	T	T	T	F	F	F
AGE	0	1	2	0	1	2
VASC_T	0.0007	0.0011	0.0015	0.00035	0.00055	0.00075
VASC_F	0.9993	0.9989	0.9985	0.99965	0.99945	0.99925

Table B.5: CPD describing the probability distribution for variable 'Vascular Lesions'.

Signs CPDs:

- Scaly Skin

BKL	T	T	F	F
AKIEC	T	F	T	F
SCA_T	0.8	0.65	0.75	0.0001
SCA_F	0.2	0.35	0.25	0.9999

Table B.6: CPD describing the probability distribution for variable 'Scaly Skin'.

- Smooth Skin

DF	T	F
SMO_T	0.75	0.2
SMO_F	0.25	0.8

Table B.7: CPD describing the probability distribution for variable 'Smooth Skin'.

- Deformed moles

MEL	T	F
MOL_T	0.9	0.003
MOL_F	0.1	0.997

Table B.8: CPD describing the probability distribution for variable 'Deformed Moles'.

- Skin Bumps

DF	T	T	F	F
BCC	T	F	T	F
BUM_T	0.98	0.98	0.40	0.01
BUM_F	0.02	0.02	0.60	0.99

Table B.9: CPD describing the probability distribution for variable 'Skin Bumps'.

- Bleeding

MEL	T	T	T	T	F	F	F	F
BCC	T	T	F	F	T	T	F	F
VASC	T	F	T	F	T	F	T	F
BLE_T	0.70	0.60	0.65	0.55	0.65	0.55	0.60	0.20
BLE_F	0.30	0.40	0.35	0.45	0.35	0.45	0.40	0.80

Table B.10: CPD describing the probability distribution for variable 'Bleeding'.

Bibliography

[1] G. Annessi, R. Bono, F. Sampogna, T. Faraggiana, and D. Abeni. Sensitivity, specificity, and diagnostic accuracy of three dermoscopic algorithmic methods in the diagnosis of doubtful melanocytic lesions: the importance of light brown structureless areas in differentiating atypical melanocytic nevi from thin melanomas. en. *J. Am. Acad. Dermatol.*, 56(5):759–767, May 2007.

[2] A. Chia, G. Moreno, A. Lim, and S. Shumack. Actinic keratoses. en. *Aust. Fam. Physician*, 36(7):539–543, July 2007.

[3] E. Collins, R. Achanta, and S. Süsstrunk. Deep feature factorization for concept discovery, June 2018. arXiv: `1806.10206 [cs.LG]`.

[4] R. Collins and N. Fenton. Bayesian network modelling for early diagnosis and prediction of endometriosis. November 2020.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: transformers for image recognition at scale, October 2020. arXiv: `2010.11929 [cs.CV]`.

[6] J. Gildenblat and contributors. Pytorch library for cam methods. `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

[7] S. Gnat, D. Łagowski, and A. Nowakiewicz. Genetic predisposition and its heredity in the context of increased prevalence of dermatophytoses. en. *Mycopathologia*, 186(2):163–176, May 2021.

[8] C. Hafner and T. Vogt. Seborrheic keratosis. en. *J. Dtsch. Dermatol. Ges.*, 6(8):664–677, August 2008.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, December 2015. arXiv: 1512.03385 [cs.CV].

[10] H. Kittler, H. Pehamberger, K. Wolff, and M. Binder. Diagnostic accuracy of dermoscopy. en. *Lancet Oncol.*, 3(3):159–165, March 2002.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, August 2017. arXiv: 1708.02002 [cs.CV].

[12] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo. Swin transformer v2: scaling up capacity and resolution, November 2021. arXiv: 2111.09883 [cs.CV].

[13] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: hierarchical vision transformer using shifted windows, March 2021. arXiv: 2103.14030 [cs.CV].

[14] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A ConvNet for the 2020s, January 2022. arXiv: 2201.03545 [cs.CV].

[15] M. Lucius, J. De All, J. A. De All, M. Belvisi, L. Radizza, M. Lanfranconi, V. Lorenzatti, and C. M. Galmarini. Deep neural frameworks improve the accuracy of general practitioners in the classification of pigmented skin lesions. en. *Diagnostics (Basel)*, 10(11):969, November 2020.

[16] A. G. Marzuka and S. E. Book. Basal cell carcinoma: pathogenesis, epidemiology, clinical features, diagnosis, histopathology, and management. en. *Yale J. Biol. Med.*, 88(2):167–179, June 2015.

[17] N. H. Matthews, F. Moustafa, N. M. Kaskas, L. Robinson-Bostom, and L. Pappas-Taffer. 41 - dermatologic toxicities of anticancer therapy. In J. E. Niederhuber, J. O. Armitage, M. B. Kastan, J. H. Doroshow, and

J. E. Tepper, editors, *Abeloff's Clinical Oncology (Sixth Edition)*, 621–648.e5. Elsevier, 2020.

[18] F. E. Myers DJ. Dermatofibroma. en. *StatPearls*, 2022.

[19] M. S. Nestor and M. B. Zarraga. The incidence of nonmelanoma skin cancers and actinic keratoses in south florida. en. *J. Clin. Aesthet. Dermatol.*, 5(4):20–24, April 2012.

[20] M. RASTRELLI, S. TROPEA, C. R. ROSSI, and M. ALAIBAC. Melanoma: epidemiology, risk factors, pathogenesis, diagnosis and classification. *In Vivo*, 28(6):1005–1011, 2014. ISSN: 0258-851X. eprint: `https://iv.iiarjournals.org/content/28/6/1005.full.pdf`. URL: `https://iv.iiarjournals.org/content/28/6/1005`.

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: visual explanations from deep networks via gradient-based localization, October 2016. arXiv: `1610.02391 [cs.CV]`.

[22] A. Shaker, M. Maaz, H. Rasheed, S. Khan, M.-H. Yang, and F. S. Khan. SwiftFormer: efficient additive attention for transformer-based real-time mobile vision applications, March 2023. arXiv: `2303.15446 [cs.CV]`.

[23] H. Tran, K. Chen, A. C. Lim, J. Jabbour, and S. Shumack. Assessing diagnostic skill in dermatology: a comparison between general practitioners and dermatologists. en. *Australas. J. Dermatol.*, 46(4):230–234, November 2005.

[24] P. Tschandl. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions, 2018.

[25] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie. ConvNeXt v2: co-designing and scaling ConvNets with masked autoencoders, January 2023. arXiv: `2301.00808 [cs.CV]`.

[26] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization, December 2015. arXiv: 1512.04150 [cs.CV].

Acknowledgements

This endeavor would not have been possible without my advisor, Prof. Samuele Salti, whose invaluable guidance was instrumental in comprehending and implementing the more recent solutions for Vision Transformers.
I am also thankful to Dr. Marzia Baldi, who generously provided her knowledge and expertise in the dermatological field.
Thanks should also go to Intré and the members of 'Gilda DermatologIA' for their meaningful contributions, guiding me in deploying the application in a real-world scenario.
Lastly, I'd acknowledge my family and friends for their unwavering support and motivation throughout this journey.