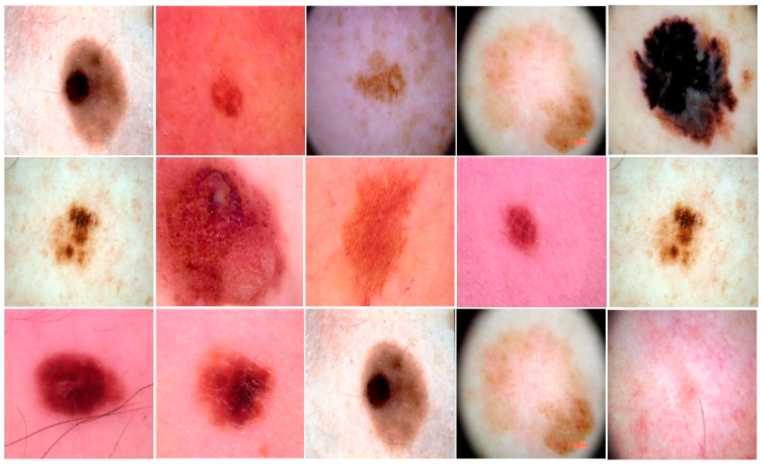
Telecom Paris

# Challenge IMA205



# Classification of dermoscopic images of skin lesions

*Elaborated by:*

**Haithem DAGHMOURA**

**Academic year :**
**2023/2024**

# Abstract

Skin lesions pose a significant public health concern due to their potential to degenerate into cancerous growths, with melanoma being particularly dangerous. Early detection and accurate classification of skin lesions are crucial for timely intervention and treatment. This report presents the methodology used for skin lesion classification challenge, where dermoscopic images were classified into eight diagnostic classes using machine learning algorithms. Notably, our model, submitted under the username **Haithem_DAGHMOURA**, achieved a **private score of 0.687**, representing a balanced accuracy score, and secured the **15th rank** on the leaderboard. We detail the feature extraction techniques employed, the classification algorithms utilized, and the evaluation metrics used to assess the performance of the classification system.

# Contents

# List of Figures

# Introduction

Skin cancer is one of the most prevalent and potentially fatal forms of cancer, with melanoma being particularly aggressive and challenging to treat. Early detection and accurate classification of skin lesions play a pivotal role in improving patient outcomes by enabling timely intervention and treatment. In recent years, there has been a growing interest in leveraging machine learning techniques for computer-aided diagnosis (CAD) in dermatology, aiming to enhance the efficiency and accuracy of skin lesion classification.

The objective of this report is to present the findings of our participation in the skin lesion classification challenge hosted on Kaggle. The challenge involved classifying dermoscopic images of skin lesions into eight distinct diagnostic classes, including melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, and squamous cell carcinoma. Leveraging a dataset of dermoscopic images with associated metadata, our task was to develop a machine learning model capable of accurately categorizing these lesions.

In this report, we provide a comprehensive overview of our approach to feature extraction, selection of classification algorithms, evaluation metric employed, and the experimental setup utilized during the challenge. Additionally, we discuss the insights gained from analyzing the results, challenges encountered, and potential avenues for future research in the domain of skin lesion classification.

# 1 Data Overview

The skin lesions in the dataset are classified into eight diagnostic classes, as outlined in the challenge description. These classes include melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, and squamous cell carcinoma. This dataset comes in form of a zip file containing two directory of several images with potentially a mask when available and 2 csv files containing the tabular metadata describing other numerical and categorical features.

## 1.1 Images

The images provided are located in the given directories with each image having its ID +'.jpg' as name and when available a segmentation associated to it named ID + '_ seg.png'.This images comes in various sizes and with different views. The labels are provided in a seprate csv file containing the ID and the metadata of each image.

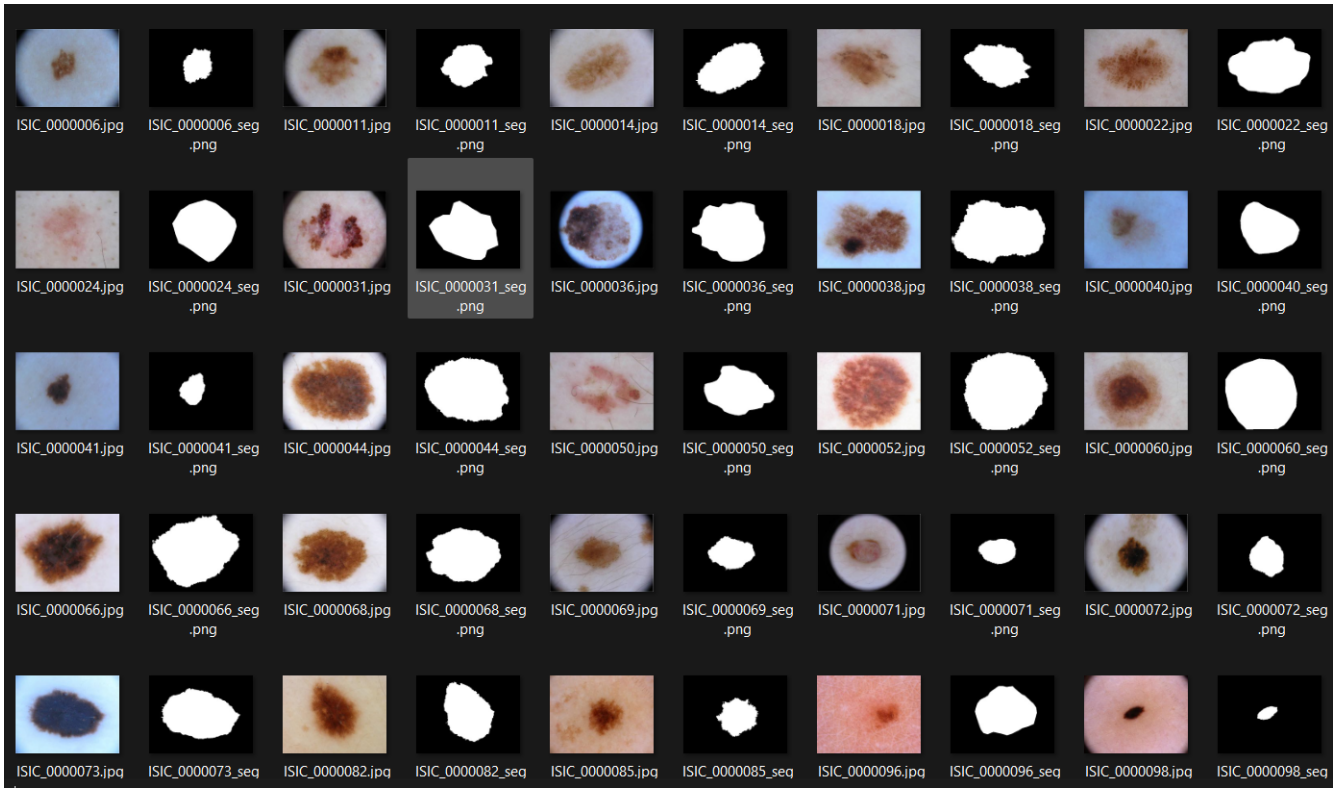



Figure 1: Overview of the images

## 1.2 MetaData

Along side the images folders 2 csv files containing other tabular data is given. This data consists of the age of patients, the gender and the location of the taken image. The gender and the location are the 2 categorical features and the age is given in numbers and thus can

be considered as continous. Finally, the metadata for a given image can contain missing values given by the value of "nan".

### 1.3 Data Imbalance

Upon examining the dataset, it was observed that the distribution of samples across the classes is imbalanced. Certain classes, such as class 6 and 7, are represented by fewer samples compared to others. This class imbalance may pose challenges during model training and evaluation, necessitating careful consideration of class weights and sampling strategies.

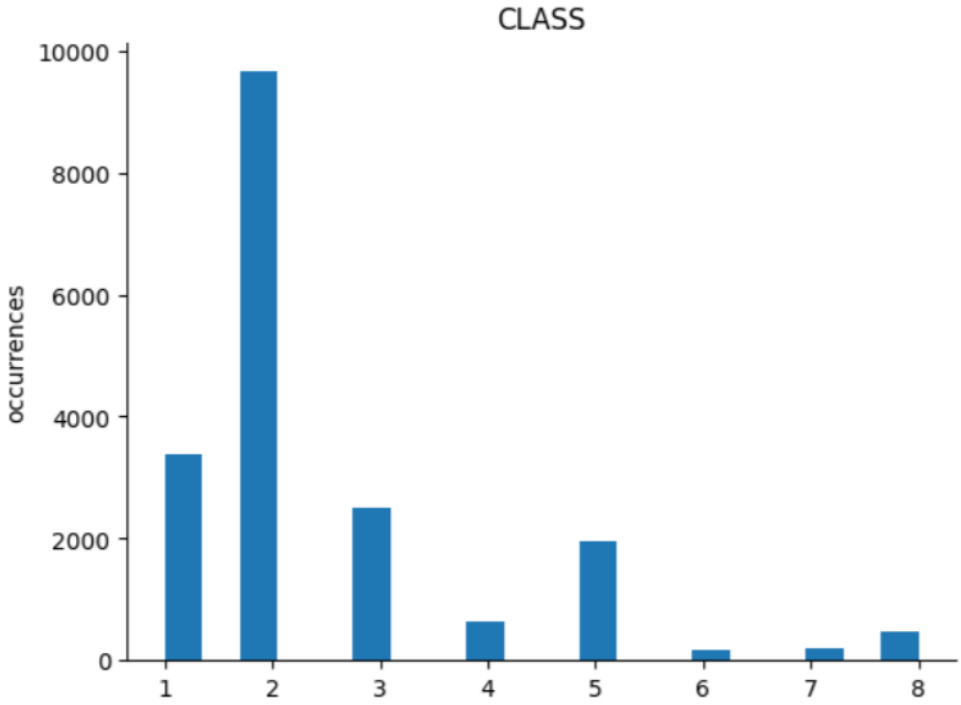



Figure 2: histogram of classes occurrences

To deal with this imbalance we used data augmentation techniques, that we will describe in the next sections, in order to over sample the underrepresented classes.

## 2 Description of the overall workflow

In this section we will be describing the general workflow used for this challenge in the same order done in the associated notebook. Each part will be explained and detailed.

Please note that for the notebook to run as it is expected to, you need to download the "ima205-challenge-2024.zip" file from the kaggle competition and put it in the same directory as the notebook provided then run it (thus the directory should only contain the zip file and the notebook).

The list of imports and the all the python libraries versions are provided in the first section of the notebook entitled "Packages version and Imports" The next section unzippes the file in the current directory. If you already have unzipped it please comment that section.

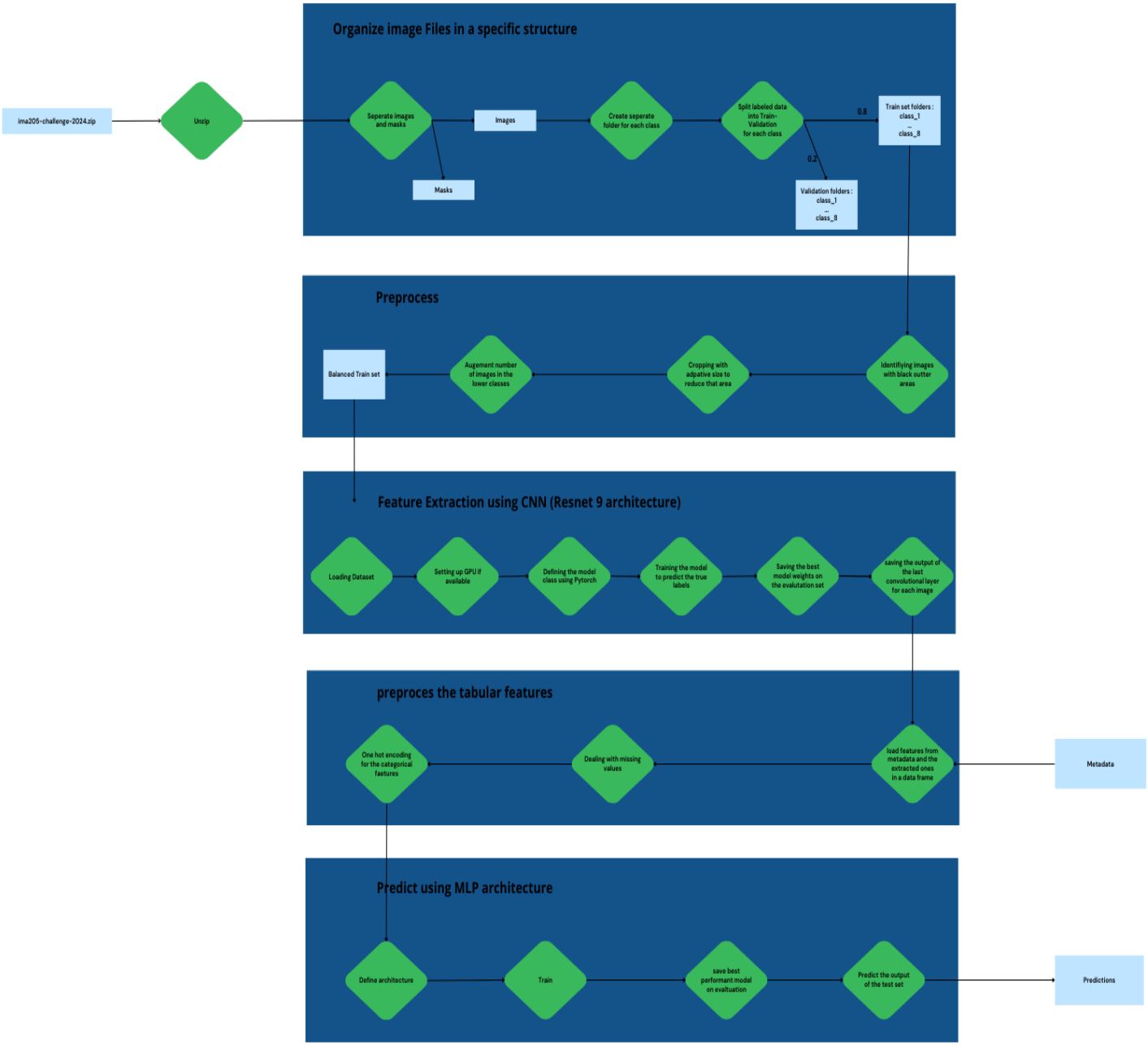



Figure 3: Overview of the general workflow

## 2.1 Organizing files

This part deals with the file structure we want to use. The provided folders contains the images and masks for every class. For things to be more structured and easier to use later on we chose to reorganize these files. First we will be moving the masks into other folders thus only the images will remain in the current file.

Afterwards we proceed by creating 8 seperate folders. Each folder will contain the image

class and will have name class_'Class'. This will be useful later on for the data seperation, augmentation and loading.

Finally we will be seperating our labeled data into Train and Validation data sets with ratio 0.8 for the train. The validation dataset will only be used afterwards for evaluation puproses and won't be included or seen in the whole process of training. This will help identify and test out the performance of the method before submitting. It is important to note that we used this proportion of splitting for each seperate folder thus we have achieved a **Stratified** splitting and making sure that the validation set is representative for our data since we have the same relative proportion of classes and the same balance between them as in the original dataset. This allows us as well to be able to use the banaced accuracy metric with weights that are given and that we got upon using the formulas described in the evalutaiton method on kaggle. In the end of this process we will end up with the follwing directory structure:

```
.
└── work_directory/
    ├── ima205-challenge-2024.zip
    ├── Haithem_DAGHMOURA.ipynb
    ├── Train/
    │   ├── class_1/
    │   │   ├── ISIC_0000002.jpg
    │   │   └── ...
    │   ├── ...
    │   └── class_8/
    │       ├── ISIC_0024329.jpg
    │       └── ...
    ├── Validation/
    │   ├── class_1/
    │   │   ├── ISIC_0000029.jpg
    │   │   └── ...
    │   ├── ...
    │   └── class_8/
    │       ├── ISIC_0024562.jpg
    │       └── ...
    ├── Test/
    │   └── Test/
    │       ├── ISIC_0000006.jpg
    │       └── ...
    ├── Test_masks/
    │   ├── ISIC_0000006_seg.png
    │   └── ...
    └── Train_masks
```

Figure 4: Final Directory organisation

## 2.2 Preprocessing

Upon further inspections of the image we noiticed that a big number has a black outter. This specific images can lead to poor training later since the images are not consistant over all the samples.

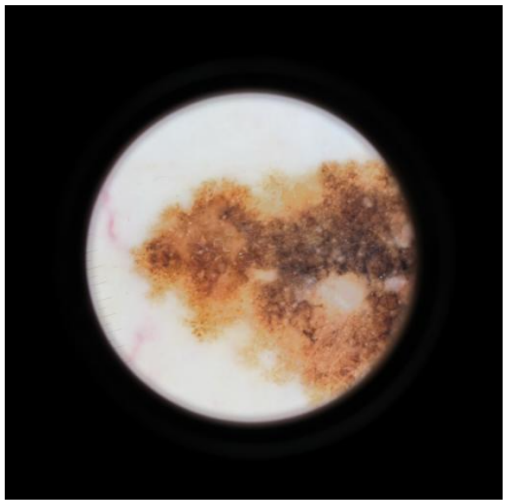


Figure 5: Example of image with black outter

In order to deal with this issue, we implemented a method that first, detects wether an image has this outter black area and then croppes it depending on the size of the actual skin image. So we iterate over all the images then test if they need to be cropped if yes we proceed.

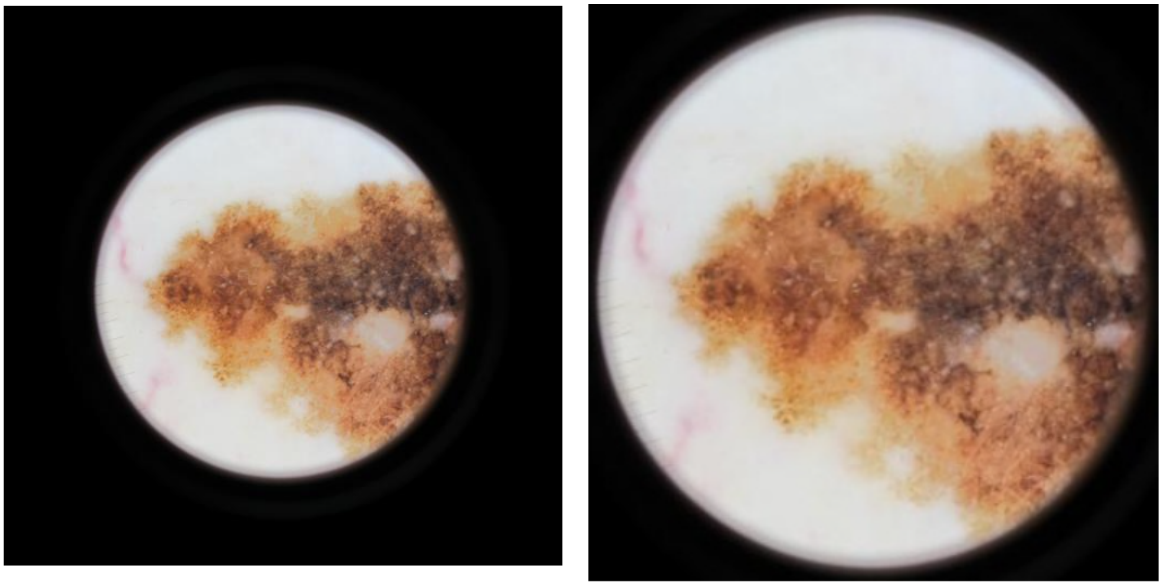
### 2.2.1 Detecting the images to crop

This method is quite forward and doesn't take alot of time. It is based on the observation that all the images that needs to be cropped has a border with very low value compared to the center. So we start by reading the image in integers, we fix a threshold = 10 and we check the top, buttom, left and right borders (with width 10) if they all have a value less than the fixed threshold and if yes than the image has to be cropped

### 2.2.2 Cropping the images with black outter

The best method to use here is cropping the unwanted areas. So we started by looking at the samples that we got and we found out that there is a consistant circular-like shape for the wanted image surrounded by the black areas. Thus the idea came to first segment the image into two regions using a threshold. Then detect the possible countours using cv2.findContours() function. Finally we search for the maximum countour using cv2.contourArea() as criteria ,then we approximate the contour of the wanted area with cv2.approxPolyDP() to lower the number of points needed to describe the shape. And we deem the shape to be circular if we need more

than 7 points to approximate it. If that is the case we procced by finding the bounding box using cv2.boundingRect() then cropping using this bounding box



(a) Original image (b) Cropped image

## 2.3 Data augmentation

Now in oder to address the unbalance issue which is a big one if we are aiming to train a convolutional neural network with alot of wieghts, We decided to generate more samples for the under representated classes. This is done only in the training folder. The validation folder will be kept as it is.

This is built upon the assemption that for a given image the type of skin disease doesn't change with spacial transfromation (espacially with isometries). For instance if we know that the image A is of class 1 then image A flipped or rotated should be as well of class 1. This is also very intuitive because a doctor, can look at the skin from different angles and take the image from different point of views and still have the same diagnosis. This enables us to generate alot of samples given the ones that we have without falling in the trap of redundancy of the data. For this part, we chose to use rotation with range=20, a width and height shift with range=0.1, a zoom range of 0.2, horizontal and verical flip, and finally a brightness transfromation with range = [0.9, 1.1]. The brighteness can be aswell justified by the diffrent light conditions that taken picture can be in. As consequence the result shouldn't depend on these transfromations

for instance if we take a look at the class 7, In the training set we have only 152 samples and we aim to generate 7573 more. This can be done using the above transformation by considering every single image and generate 7 images from each one.

(a) Image 1 (b) Image 2 (c) Image 3

(d) Image 4 (e) Image 5 (f) Image 6

(g) original image

Figure 7: data augmentation example

As we can see above the generated images are quite realistic and should have the same label.

## 2.4 Feature extracting using CNN

Now after preparing the direcotory and preprocessing the images, we are ready to proceed with the feature extracion. The goal of this part is to get a set of relevant features describing the relation between the image and the labels. So, first we start by loading the Training and validation dataset then prepare useful functions to utilize the GPU computing and check if it is available. Afterwards, we chose to use a CNN in particular Resnet 9 architecture. Thus we define the model using pytorch classes inhertance, alongside other methods that we deem useful for evaluation and the training loop.

### 2.4.1 ResNet-9 Architecture

ResNet-9 is a variant of the ResNet (Residual Network) architecture, which is a type of deep convolutional neural network (CNN) designed to address the vanishing gradient problem during training of very deep networks.

- **Input Layer**: The input layer accepts the input image data with (128,128) as size.
- **Convolutional Layers**: These layers starts with a series of convolutional layer followed by batch normalization and ReLU activation functions. These convolutional layers perform feature extraction from the input image.
- **Residual Blocks**: The core idea behind ResNet architecture is the use of residual blocks. A residual block contains skip connections (or shortcuts) that allow the model to learn residual mappings instead of directly fitting the desired underlying mapping. ResNet-9 consists of multiple residual blocks. Each residual block typically contains two convolutional layers with batch normalization and ReLU activation, followed by a skip connection. The skip connection adds the input of the block to its output, allowing the gradients to flow directly through the network without vanishing.
- **Pooling Layers**: After the residual blocks, max pooling is applied to downsample the feature maps and reduce their spatial dimensions.
- **Fully Connected Layers**: Finally, ResNet-9 ends with flatten operator followed by a dropout layer (we chose rate =0.2) followed by one fully connected layer followed by a softmax layer for classification tasks. This fully connected layer aggregate the features learned from the convolutional layers and make predictions based on them.

### 2.4.2 Output Layer

The output layer produces the final predictions, typically representing the probability distribution over the classes in a classification task. We used this prediction in order to evalue our feature extraction capabilities on the validation dataset. The loss function in training is the categorical loss entropy since all classes have approximatly the same number of samples. For the validation and model selection we used weighted accuracy since we did a stratified split on the original data so we have the unbalanced classes issue.

ResNet-9 is a relatively shallow variant of the ResNet architecture, consisting of fewer layers compared to deeper variants like ResNet-50 or ResNet-101. Despite its simplicity, ResNet-9 can still achieve good performance on various image classification tasks, especially when training data is limited or computational resources are constrained which is why we chose this architecture for feature extraction. In addition, upon reading the paper intiteled "Conditional

dependence tests reveal the usage of ABCD rule features and bias variables in automatic skin lesion classification"[2] we understood that deep learning models that were selected incorporate asymmetry, border and dermoscopic structures in their decisions. So we decided to only use the CNN for feature extraction which in tern will implicitly incorporate the ABCD rules.

### 2.4.3 Training ResNet-9 and results

The trainig process is based on forward-backward process. after choosing adam optimizer and the batch size equal to 128 we proceed in a loop for training and we update each time the parameters using the gradient. The chosen loss function was the weighted categorical loss function. At the end of each epoch we compute the train loss, validation loss and the weighted validation accuracy and save the obtained weights if we increased the weighted validation accuracy allowing us to have the best generalisation model. The best model achieved a **validation accuracy** on unseen data equal to **0.659** which is consistant with the result of the first submission on kaggle achieving **public score of 0.648**.

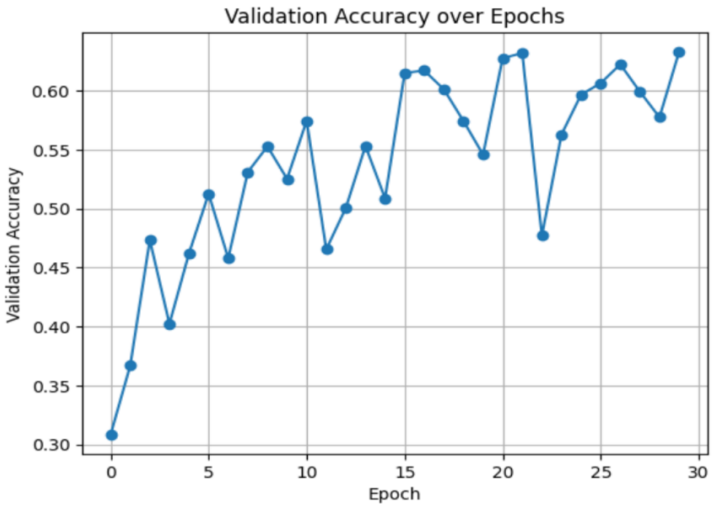



Figure 8: Validation accuracy over epochs

### 2.4.4 Feature Extraction from images

Now that we have a model that can decently predict the output, we are quite sure that its capability to extract features are good. So we proceed by predicting the ouput of the forward pass while ignoring the last fully connected layer and the dropout operation. This enables us to get 512 numerical features given an image of size 128x128. We then save the results in an intermediate csv file that we will be using later on for the final perdiction. It is important to note that we need to do this transformation on the training and validation images aswell as the test images in order to get a new set of features for further supervised learning approach where we will be combining them with the metadata that we still hasn't used so far

## 2.5 Working with metadata

The aim of this section is to combine the obtained features so far with the ones given to us in the metadata files in order to build a Multilayer Perceptron able to further increase our score of classification. To do so, we start by reading the metadata file and the features file that we got from the pervious cells in dataframes then we merge them using the ID column. This enables us to get the desired dataframe where each column contains the ID the CLASS and the different features of each sample seperatly on the Train and the validation sets.

### 2.5.1 Dealing with missing values

Upon inspecting the metadata we noitice that we have a number of missing values. For the Training and validation we have got 284 nan in feature "SEX", 324 in feature "AGE" and 1970 in feature "POSITION" with more than 1500 missing value in the most frequent class (containing more than 9000 samples) . Given that we have 18 998 samples overall we deemed that replacing the missing values with good enough ones won't affect the training process. For the "SEX" we replaced it with "male" which is the most commun value in the dataset. For "AGE" we replaced the missing values by the the most frequent age in that category. And finally for the "POSITION" we replaced it with the most frequent value over all the dataset which was 'anterior torso'.

Then we realised that the age wasn't actually taking continous values but rather a finite set of discrete integers further more after visualizing the boxplot it was clear that the value of age itself doesn't afftect the results as much as the range of age it belongs to. So we decided to transform it from numerical to categorical by grouping the ages in sets of intervals with range 5.

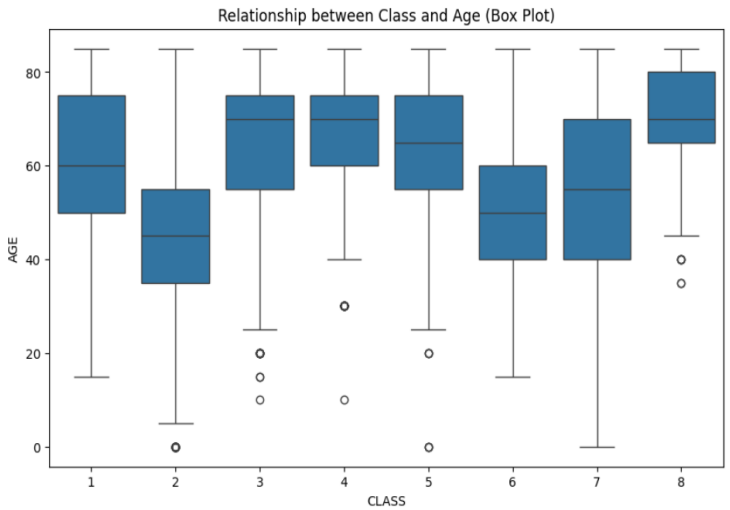



Figure 9: Boxplot of age for each class

The process was the same for the test set where we chose the values that we replaced with

in the training set and applied them in the test set.

### 2.5.2 Encoding categorical features

Now the last thing left to do before training is transforming the categorical features into numerical ones. To do this we chose the one hot encoding technique. In fact, this is perfect for our case since the categorical features we have in our dataset don't take alot of differnt values. Furthermore in order to ensure that we get the same one hot encoding we merged the Train, validation and test data frames, done the encoding then splitted them.

## 2.6 Generating prediction using MLP and Obtained results

Now after obtaining the features from images and cleaning the metadata we are ready to do the predictions. For this end, we chose to use Multilayer Perceptron with 3 hidden layers each one followed by a Relu function, a batch normalization and Dopout in order to increase our generalisation capabilities. In fact, we used MLP because it is a very powerful tool. In theory MLP can approximate any continous function provided that we give it enough number of neurons.

After setting up the model we proceeded by the training loop. The loss function was the weighted categorical loss entropy. Then, we visualize the performance in each epoch using the blanced accuracy mertric with the weights that we computed using the provided formula. The training was conducted using mini batches and used the adam optimier for the weights updates.

Finally we evaluated the results on the validation set where we got a **balanced accuracy of 0.684** and after submission on kaggle we got a **public score of 0.703** and a **private score of 0.687** which was within our expectations.

The final step consisted in generating the output using the obtained features and the trained model on the test set. The file to be generated is named "predictions_final_to_submit.csv" and this is the last work to be subimitted.

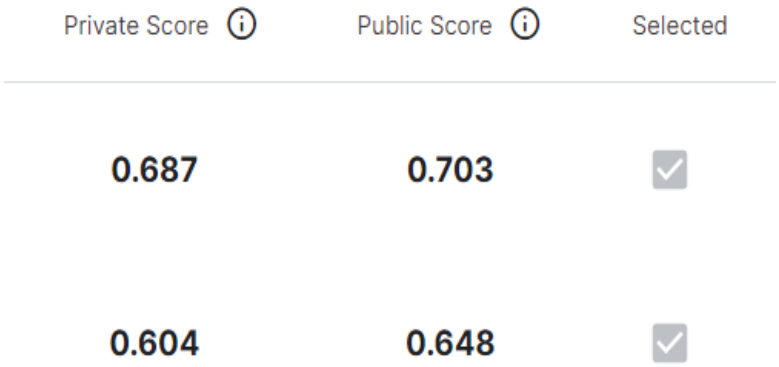



Figure 10: Final submission results on kaggle

# 3 Discussion

## 3.1 Method used

In this work, various techinques were utilizd in a complex way to achieved in our opinion good and satisfiying results. The method used was inspired from several resources. The idea of dealing with the outter black area was used and recommanded by the winners of a similar challenge on ISIC 2019 Challenge in their paper [1]. Further more using a CNN architecture as feature extractor and utilizing the ouput by adding to the meta data and using them as input to a MLP was also inspired from their work. The performance of this method on the submission done in kaggle was also within the our expectation. In fact, we used a validation set with a stratifying split to evaluate the model performance and chose the best generalizing one.

## 3.2 Potential Enhancements

Finally, We want to mention that instead of using ResNet9 as feature extraction, fine tuning a pretained model with more complexity yieleded for us a much higher accuracy reaching approximatly 0.84 on the validation set (by using pretrained ResNet-50 as a starting point). However, we chose to not use it because in the discription provided in kaggle it was explicitly mentioned that using extrior dataset out of the scope of the challenge was prohibited. Thus we understood that, by extension, using a model that was trained on such a dataset would be considered as "cheating". As consequence we preferred being restricted to only the dataset that we have and built and trained a smaller architecture from scratch (ResNet9).

# Conclusion

In conclusion, this project aimed to tackle the challenge of classifying dermoscopic images of skin lesions into eight diagnostic classes using machine learning algorithms.

Despite encountering challenges such as imbalanced data and varying image qualities, our model demonstrated promising results.

Overall, our efforts yielded a competitive performance, as evidenced by our ranking and balanced accuracy score on the leaderboard. However, there's always room for improvement. Future work could focus on incorporating more sophisticated feature extraction techniques and using larger pre-trained models to further enhance classification accuracy.

This project underlines the potential of machine learning in dermatology, particularly in aiding early detection and diagnosis of skin lesions. By harnessing the power of data-driven approaches, we can contribute to the advancement of non-invasive computer-aided diagnosis systems, ultimately improving patient outcomes in dermatological healthcare.

# References

[1] Nils Gessert, Maximilian Nielsen, Mohsin Shaikh, René Werner, and Alexander Schlaefera "Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data" Published online 2020 Mar 19

[2] Christian Reimers, Niklas Penzel, Paul Bodesheim, Jakob Runge, Joachim Denzler, "pendence tests reveal the usage of ABCD rule features and bias variables in automatic skin lesion classification" published in 2021