I. Introduction:

A lot of people sadly suffer from skin diseases globally that not only cause physical discomfort but also mental stress (shame, embarrassment, etc.). That is why early and accurate classification of these conditions might be consequential for a more effective treatment. Traditionally, the diagnoses were made by visual examination from the doctor, sometimes complimented by biopsies and other tests. But this approach is subjective and can be unreliable, especially for complex and less common conditions.

The goal of this project is to address the limitations of traditional diagnosis methods by using deep learning tools and techniques that have already proved their worth in image analysis tasks, including medical image classification. By putting deep learning to use, we are going to develop a model that can classify skin diseases, to aid dermatologists in their diagnoses and ultimately improve patient care and satisfaction.

II. Different approaches:

All the different models used the same set of libraries that we are going to enumerate here: pandas, sklearn, requests, TensorFlow, seaborn, NumPy, matplotlib, cv2, keras, pickle, imageio, random, and spicy.

Before creating the models, we first opted for downloading and organizing the images into folders to make the following tasks easier and faster due to them being in a local space. We started by creating 3 folders for each type of data(train/test/validation) which then got 3 sub-folders each. The number and name of these sub-folders has been scrapped from the csv “`"fitzpatrick17k.csv"` which resulted in three sub-folders each containing specific disease names to be predicted next: *benign*, *malignant*, *non-neoplastic*. The model first predicts from the first 3 categories (benign/malignant/non-neoplastic) and then makes another prediction which type of disease it is.

Another important step is the splitting of the dataset. The approach taken maintains the class distribution within each split (train, validation, test) by proportionally distributing the images based on the split ratio 60-20-20.

Last but not least, before getting to the bulk of the models, some data pre-processing proves to be necessary. In short, we read the picture files and load each image file into an array in which each element corresponds to a pixel with 3 channels with an integer from 0 to 255.

- Reading the picture files
- Decoding the JPEG content to RGB grids of pixels
- Converting the pixels into floating-point tensors

- Rescaling the pixel values (between 0 and 255) to the [0,1] interval because neural networks prefer dealing with smaller input values.

a. **Two-step model:**

Our optimal approach is the culmination of numerous iterations of techniques, which we will detail in the subsequent sections. The two-step model (Fig. 1) for classification begins with an image input, proceeding through two distinct steps. The initial step involves a model that classifies images into one of three categories: Malignant, benign, or non-neoplastic. Then in the second step for each category, we developed three separate models that further classify the images into one of our 114 sub-categories. The reason behind this methodology is that models designed with fewer categories tend to be less complex. For example, in our CNN, having fewer categories means the final layer will have fewer outputs, potentially making the model's predictions more robust.

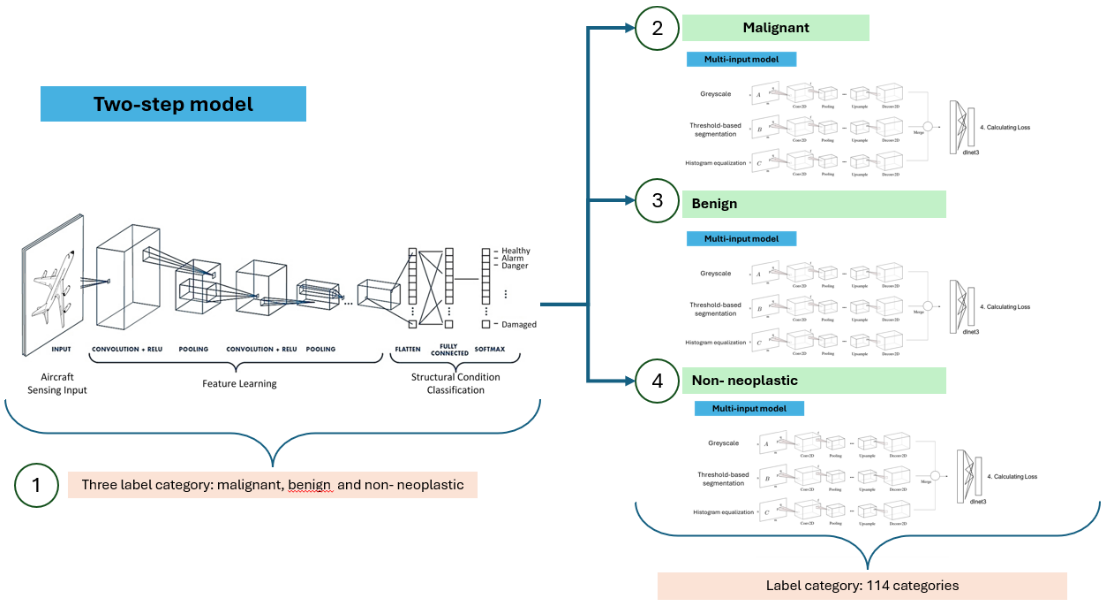


Fig.1: Diagram explaining the two-step model

Our first step is a traditional CNN model in which it takes as an input an image that is then passed through 2 types of layers (hence the name): Feature learning and Classification. In the second step of our approach, we implemented a multi-input model with three different image preprocessing techniques: Histogram Equalization, Otsu Thresholding, and Grayscale Conversion (each of these techniques will be explained in more detail in the following section).

After applying these techniques, we can finally create and run the model. It is important to note that we will primarily focus on the F1 macro score as our metric of choice because it is particularly effective in imbalanced datasets where traditional accuracy might be misleading. However, we will also present both metrics to provide a comprehensive view of our models' performance. So, these are the performances on the test dataset of the two-step model:

Accuracy	F1 Score
10.1%	8.1%

b. <u>Multi-input model:</u>

The multi-input model (*Fig. 8*), which is the second step of our previous model, is a neural network architecture that can take multiple data sources as input and process them jointly to make predictions or classifications. The benefits of these types of models are that they can leverage complementary information from different data sources which can improve the accuracy of the predictions compared to models using single inputs.

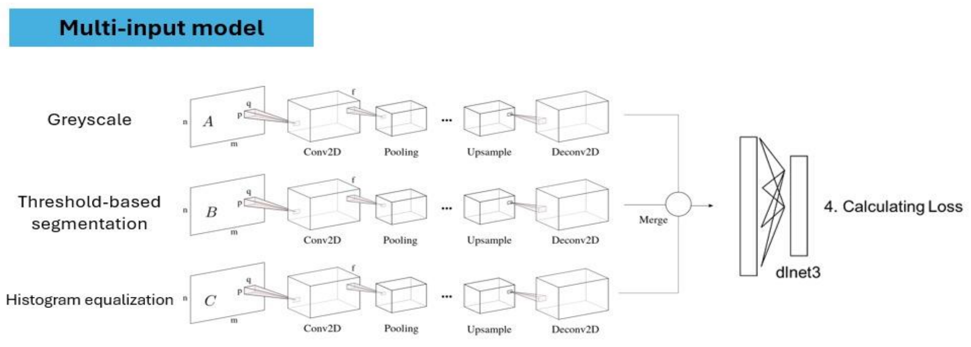


Fig.8: Diagram explaining the two-step model

Each of these inputs is the result of the following image preprocessing:

- **Histogram Equalization:** Technique used to enhance the contrast of an image by redistributing the pixel intensities; it stretches the histogram and creates a more uniform distribution. Histogram equalization makes details/features more visible and is useful for highlighting features with uneven lighting (*Fig. 2*).
- **Otsu Thresholding:** Automatic image segmentation technique that separates an image into foreground and background/black and white based on the grayscale histogram. It tries to find the optimal threshold value that maximizes the variance between the two which creates a clear distinction. It is effective for image segmentation (*Fig. 3*).
- **Grayscale Conversion:** Process of transforming a colored image into an image that only uses shades of gray. It is beneficial because it reduces data complexity by removing color

information and can be useful if the task focuses on more shapes rather than color (*Fig. 4)*.

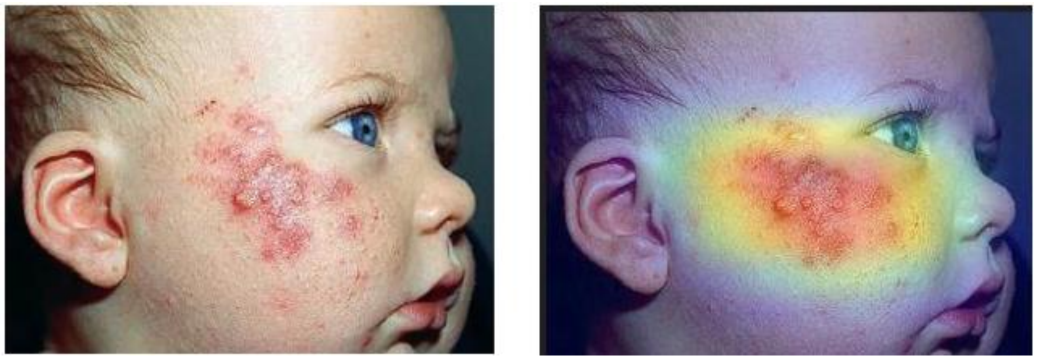
In summary, the model is a Multi-Input CNN (Convolutional Neural Network) model that consists of:

- 3 Input Layers: Grayscale, Threshold-based segmentation, and histogram equalization previously mentioned.
- Preprocessing layers: *resize and rescale* and *data augmentation*
- Shared Layers: conv2D, MaxPooling2D, and Flatten.
- Merge Layer: concatenate
- Fully Connected Layers: Dense, Batch Normalization, Dropout, and Dense (Output Layer)

The results (*Fig. 9*) for this model are as follows for the test dataset:

Accuracy	F1 Score	Precision	Recall
9.5%	8.7%	10.7%	9.5%

To enhance our understanding of the results, we implemented the Grad-CAM technique to highlight the significance of interpretability in our CNN models. This technique generates a heatmap that emphasizes critical regions within an image. For instance, in the case of acne, the heatmap accurately captures the area affected by the skin condition.



c. **Pre-trained model in Keras:**

In order to have a reference point for our results we implemented a pre-trained model MobileNetV2 from Keras which is a powerful feature extractor that has learned to extract generic features (edges, shapes, or textures) from images. The advantage of using this type of model is that you can leverage the knowledge it has accumulated from a large quantity of data (Fig.6 and Fig. 7). For this model, although we do not obtain significantly better metrics, it is important to highlight that it does reduce the model's overfitting. Thus, it would be interesting for future steps to understand the MobileNetV2 model from keras to implement it in our model to reduce overfitting.

Dataset	Accuracy
Train	48.4%
Validation	10.3%

III. <u>Conclusion:</u>

During this project, we created three image classification models: a two-step CNN model, a multi-input model, and the pre-trained Keras model. For the choice of the winning approach, we excluded the results from the pre-trained Keras model as it was a reference point. The evaluation metrics (accuracy, precision, recall, and F1 score) consistently demonstrated the superiority of the two-step model as it has achieved better performance on the test dataset and the predictions could be more robust.

However, the model was definitely not perfect and faced three main issues. Firstly, the performance, although better than its competitors', was not amazing and we could benefit from more training. Secondly, the error rate was considerably high as the model only predicted 781 right answers and 2620 wrong ones in its first prediction. This is particularly concerning in a health context, where a false negative could pose serious risks. Lastly, we encountered a critical issue of overfitting (Fig. 9) that needs to be addressed to enhance the model's generalizability and effectiveness.

There also exists two types of improvement we can use for future works; YOLO (You Only Look Once) Algorithm, and Residual Networks (ResNets). The YOLO algorithm is a single stage object detection algorithm known for its speed and real-time capabilities. Although image classification is not its primary use case, we can potentially leverage transfer learning. This approach could be interesting if we not only need to classify objects but also understand their spatial relationships within the image (which might be good for some skin diseases). As for the residual networks, they are a type of CNN architecture that takes advantage of skip connections to address the vanishing gradient problem, leading to improved performance in deeper networks. Although in our case the two-step model avoids the vanishing gradient problem by separating feature extraction and classification, a comparison in terms of performance might be worthwhile.

While the two-step model might be a good starting point, future exploration of YOLO and residual network architectures would be good venues for improvement that can lead to a more robust classification model.

Appendix

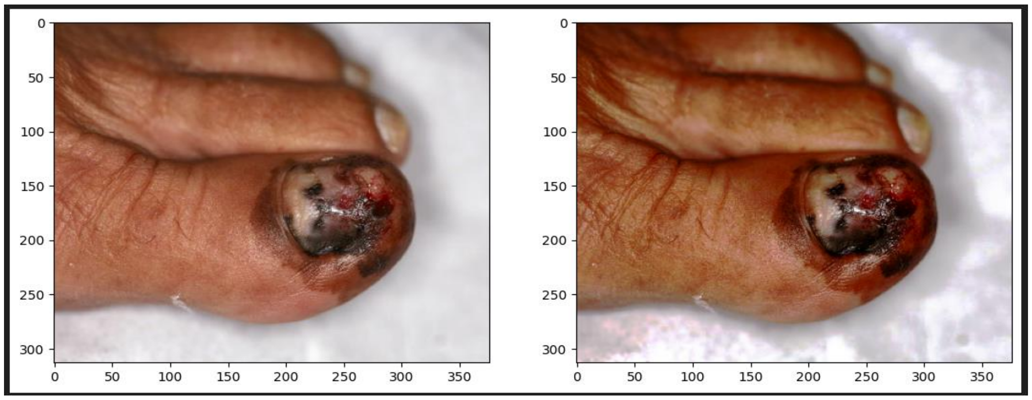


Fig.2: Image before and after applying the histogram equalization

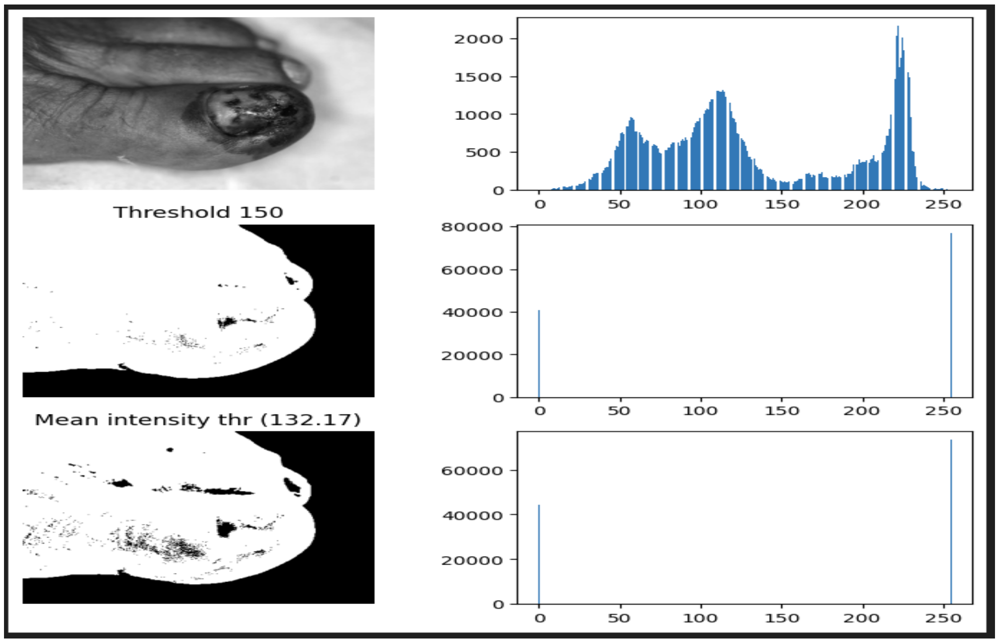


Fig.3: Image showing the effectiveness of the Otsu thresholding in separating background and foreground

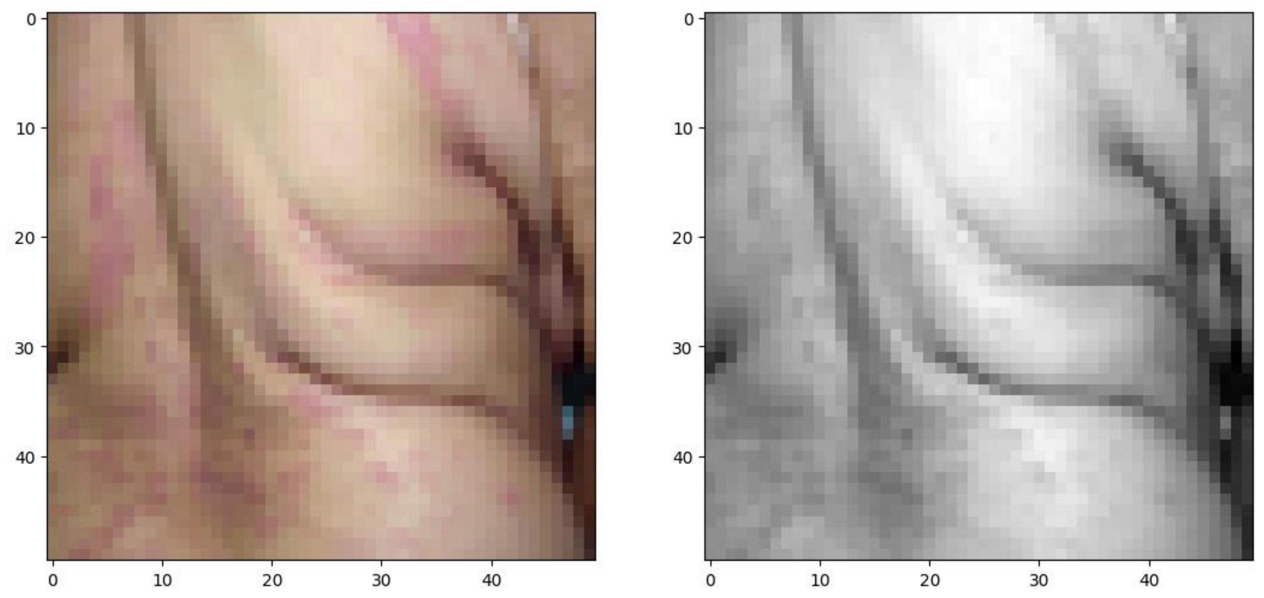


Fig. 4: Image before and after the application of the grayscale conversion

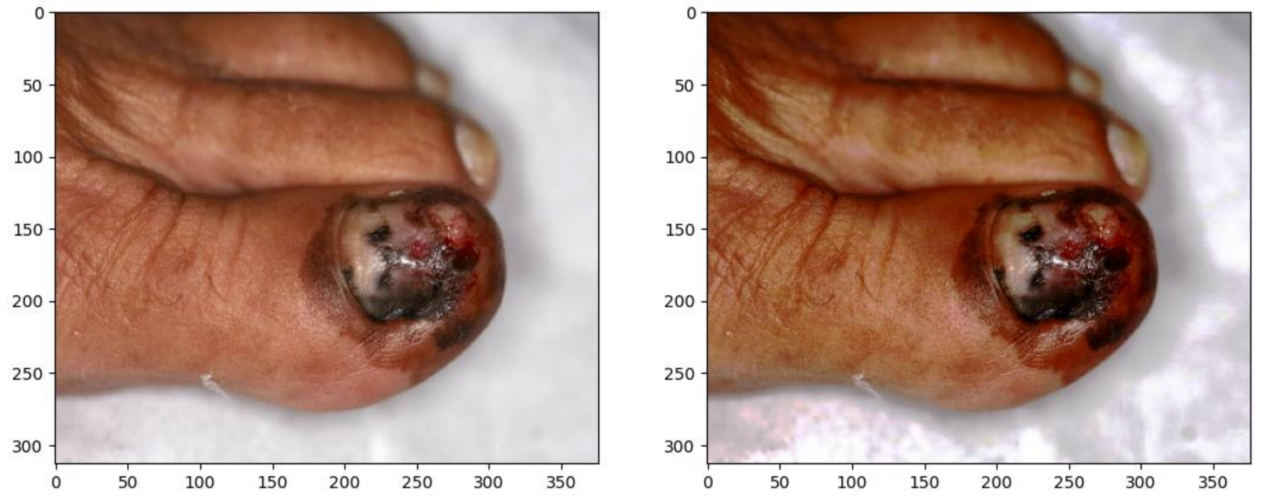
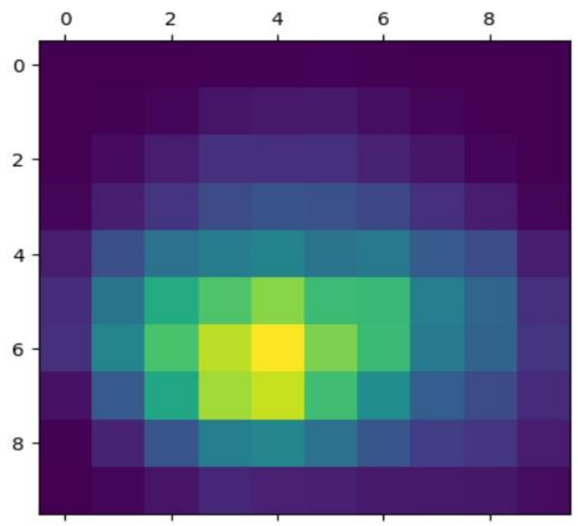


Fig. 5: Image before and after the application of the color stretching method



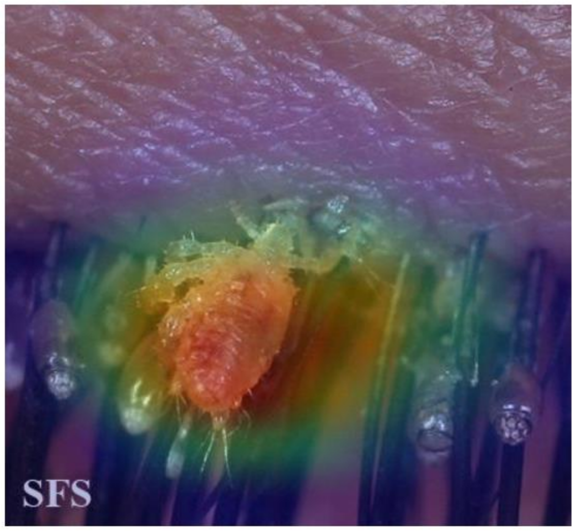


Fig. 6: Matrix showing the pixels that the pre-trained model deemed interesting

Fig.7: Image showing the focus of the model

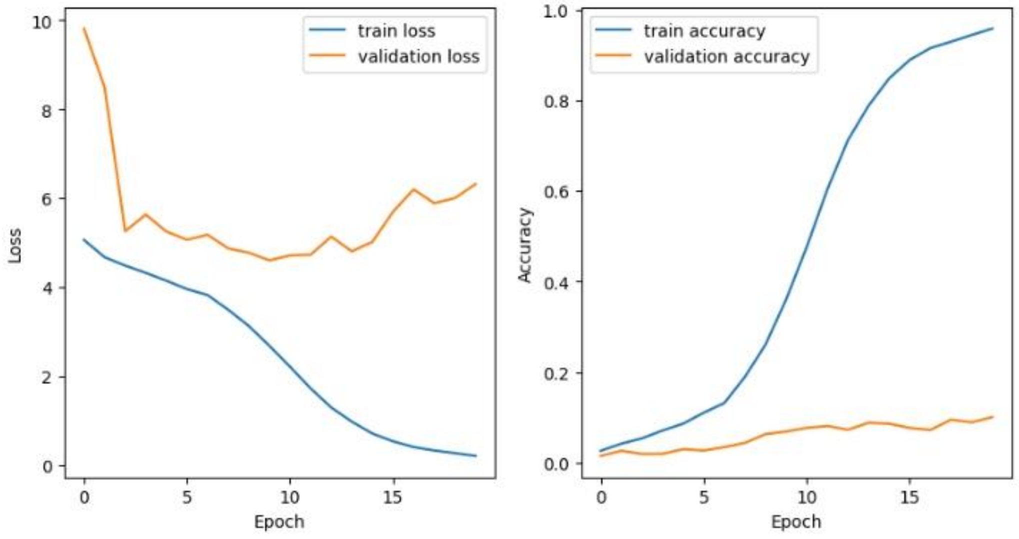


Fig. 9: Loss and Accuracy function for the multi-input model

	Model 1	*Model 2*	*Model 3*	*Model 4*
loss	*0.711*	*2.826*	*3.135*	*1.963*
accuracy	*0.737*	*0.172*	*0.221*	*0.309*
precision	*0.757*	*0.541*	*0.670*	*0.500*
recall	*0.711*	*0.006*	*0.049*	*0.041*
f1_m	*0.733*	*0.012*	*0.089*	*0.070*
f1_weighted	*0.394*	*0.912*	*0.882*	*0.812*
val_loss	*0.709*	*2.810*	*3.771*	*2.026*

val_accuracy	*0.738*	*0.190*	*0.134*	*0.309*
val_precision	*0.753*	*0.500*	*0.539*	*0.400*
val_recall	*0.723*	*0.002*	*0.017*	*0.026*
val_f1_m	*0.737*	*0.005*	*0.034*	*0.048*
val_f1_weighted	*0.378*	*0.919*	*0.929*	*0.811*
test_accuracy	*0.23*	*0.012*	*0.005*	*0.090*
test_f1	*0.21*	*0.008*	*0.006*	*0.078*

Table 1: Two-step model without grayscale and heatmap

	Model 1	*Model 2*	*Model 3*	*Model 4*
loss	*0.728*	*2.804*	*3.615*	*1.921*
accuracy	*0.735*	*0.178*	*0.124*	*0.325*
precision	*0.750*	*0.450*	*0.654*	*0.536*
recall	*0.711*	*0.014*	*0.009*	*0.054*
f1_m	*0.730*	*0.027*	*0.019*	*0.091*
f1_weighted	*0.404*	*0.909*	*0.943*	*0.792*
val_loss	*0.735*	*2.853*	*3.761*	*2.004*
val_accuracy	*0.737*	*0.161*	*0.095*	*0.273*
val_precision	*0.746*	-	*0.700*	*0.571*
val_recall	*0.721*	-	*0.009*	*0.058*
val_f1_m	*0.733*	-	*0.017*	*0.048*
val_f1_weighted	*0.389*	*0.923*	*0.948*	*0.814*
test_accuracy	*0.604*	*0.0*	*0.010*	*0.020*
test_f1	*0.337*	*0.0*	*0.009*	*0.020*

Table 2: Two-step model with grayscale and heatmap