Dermatological classification problem

Deep Learning Group Project

Group 1

Mikołaj Jarząbkowski | 20230529

Rodrigo Silva | 20230536

Nicolau Dulea | 20230544

Joana Gonçalves | 20230977

Tiago Fernandes | 20230988

Master's Degree in Data Science and Advanced Analytics

2023/202

Table of Contents

1. Introduction

In this project, a deep-learning model was developed to address a dermatology classification problem. For that, images from the "FITZPATRICK17" repository were used. The Fitzpatrick scale is a numerical classification schema for human skin colour to estimate the response of different types of skin to ultraviolet light.

The dataset used in this project contains several images, each corresponding to only one specific disease, with a total of 114 skin conditions. It is a combination of two dermatology datasets, "DermaAmin" and "ATLAS Dermatologico". The url for these images is contained in a csv file along with other information.

Several models were tested, including single input, multi-input and pre-trained models. Our chosen model was the multi-input, using both images and features from tabular data.

2. Data exploration and pre-processing

2.1. Data exploration

The metadata for the provided csv file is detailed in annex 1. For data exploration, the Pandas and NumPy libraries were used. From the provided dataset, it is possible to observe that each label has at least 53 images and a maximum of 653 images per skin condition, representing a greatly imbalanced dataset.

2.2. Data preparation

Initially, the *md5hash* and *url_alphanum* features were dropped since they did not provide relevant information. Although *url_alphanum* could have been used to extract the images that did not have a valid link in the *url* column, we decided to not use it due to the different patterns in urls and the low number of instances where this occurred. Despite the 97% of missing values present in *qc*, the remaining 3% of this feature was used for data augmentation purposes, only being dropped after this process. However, the instances marked as not correctly classified ("Potentially", "Characteristic", "Wrongly labelled" and "Other") were immediately set aside.

Then, the DataFrame was splited in three parts using the Hold-out method: train (70%), validation (15%) and test (15%). Since the dataset is heavily imbalanced, the stratify parameter from train_test_split was used. After the split, the images were extracted into three corresponding different folders. In each of the folders, the images were stored in sub-folders, named after their label. Lastly, the *url* column was dropped as well as the rows where the images did not download (ten instances). The information about the images in each folder is detailed in annex 2.

For the multi-input model, we conducted additional preprocessing steps. First, we transformed the images into arrays and resized them to a 96x96 pixels format. Additionally, images with

the .png extension were converted to .npz format, offering benefits such as improved input/output performance, reduced storage space, and faster data loading, especially for large datasets. The *label* feature was also label-encoded.

2.3. Image processing and data augmentation

Considering the provided imbalance dataset, three options were considered for data augmentation: randomly choosing images to use as base to create more images; favouring the correctly classified images; or a mix of both approaches. We chose to use the third option: on the one hand, we are training the model with variations of images that we know are correctly classified. The feature *qc* was used in this to filter the correctly classified images ("Diagnostic") in the training set. On the other hand, and since there are not many correctly classified instances, there is a need to be cautious with model generalisation.

Therefore, the responsibility of the data augmentation was divided: for the correctly classified images, 15 more images were created for each original in each label. For the remaining images, the augmentation proportion was calculated so that every disease class had approximately the same number of images (annex 3).

The data augmentation parameters were based on a random search that consisted of implementing a diverse range of augmentation techniques to enrich our training dataset, including random rotation, image shifting (both horizontally and vertically), flipping, and zooming.

In the case of image shifting, rotation and zooming, these are very common parameters when it comes to medical classification models (example: Harahap & Zulkifl, 2023). Moreover, looking at it from a pragmatic point of view, photos of medical cases/diseases will often not look the same (they may be slightly rotated, moved or zoomed in/out). We opted for +-50% zoom, since any larger value made the model performance worse. For the reflect mode, we had four options to choose from. In the end 'reflect mode' significantly outperformed the other fill modes. While shear range was explored, it did not contribute significantly to model improvement and was thus excluded. Similarly, despite initial optimism, the brightness range parameter did not enhance model performance and was omitted from the final augmentation pipeline. Examples of the obtained images can be seen on annex 4.

3. Experimental setup

Our strategy was to firstly test individually several parameters. Then, the parameters that provided better results would be combined. Finally, the knowledge grasped from these models was applied to the pre-trained models. In parallel, a multi-input model would also be tested. The conditions tested for these several phases are described in the figure 1. Due to F1 score

implementation difficulties, the results were mainly evaluated in terms of loss and accuracy for the validation dataset during training.

For modelling purposes, the Keras library and its several modules were always used.

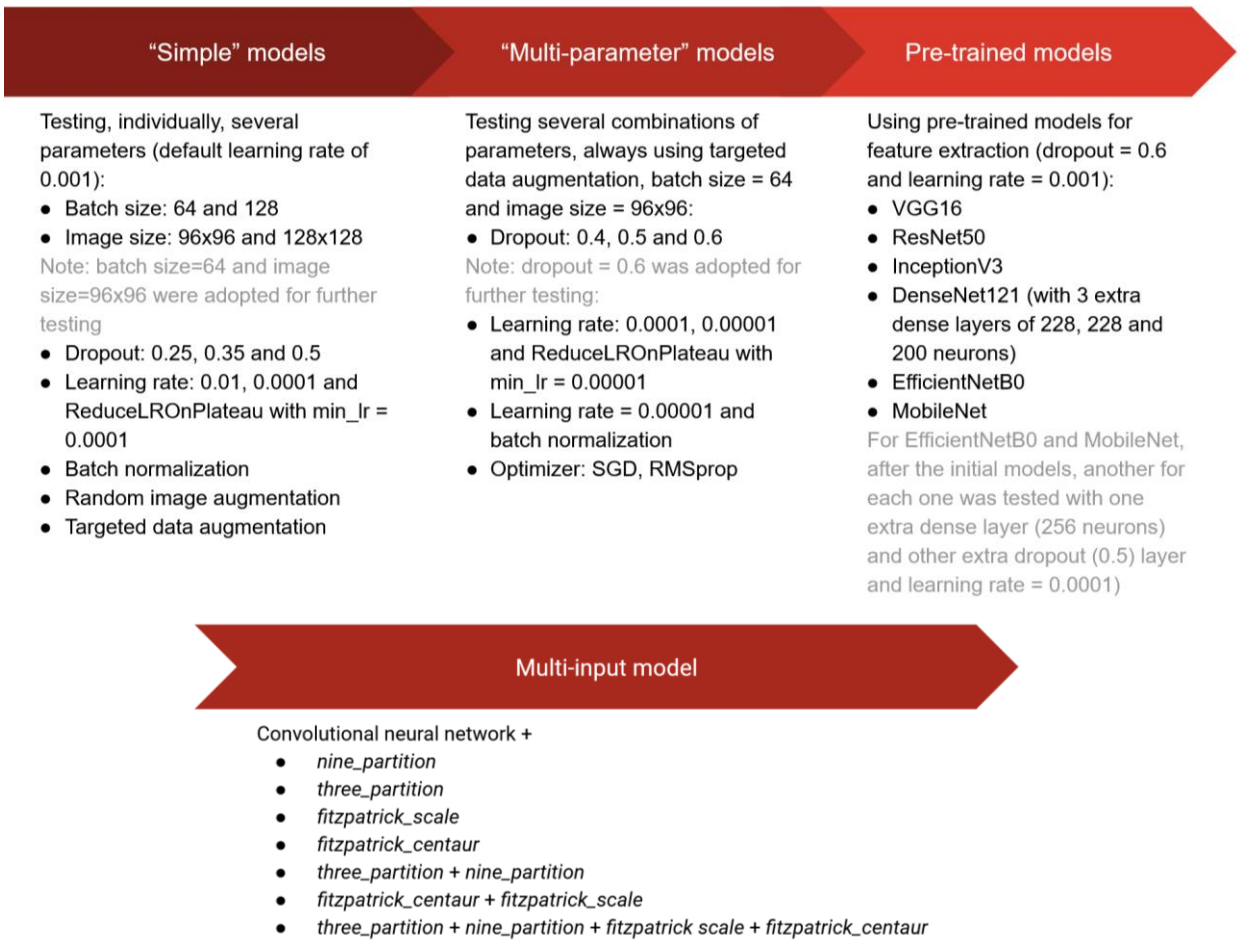

Figure 1. Modelling schema, detailing parameters and/or conditions used for the several models.

For the "simple" models, after some initial pre-testing, we defined that they would have 3 convolutional layers with filters [16, 32, 48] and ReLu as the activation function, each followed by a pooling layer (max, 2x2). Then, a flatten layer, a hidden dense layer with 228 and a final dense layer with 114 neurons and a softmax activation function. The default optimizer used was "Adam" and the learning rate was 0.0001, with "sparse_categorical_crossentropy" as loss. The training process ran for 20 epochs, with an EarlyStopping callback implemented for early stopping, using a patience of 5 epochs. Due to computational costs and the logic that these "simple" models were just initial models for testing, the image size was defined for 96x96, using a batch size of 64 pictures. Parameters important for overfit control were individually tested, such as dropout, learning rate, batch normalisation and use of random/not-random data augmentation.

With the results obtained from these “simple” models, we decided to adopt the balanced data augmentation for all models. For the “multi-parameter” models, along with the implementation of the CNN classifier, we used the RandomSearch Tuner to explore the optimal combination of hyperparameters, intending to improve the model's performance. Despite our efforts to identify the best configuration, we chose not to proceed with the best combination due to the suboptimal results obtained. After testing several combinations, the best one had a dropout of 0.6 and a learning rate of 0.0001.

Different pre-trained models were chosen based on literature use-cases and Keras performances reference: VGG16 (Groh et al., 2021), ResNet50 (Brinker et al., 2019), InceptionV3 (Tschandl, Rosendahl & Kittler, 2019), DenseNet121 (Codella et al., 2018), EfficientNetB0 (Koga et al., 2020) and MobileNet (Haenssle et al., 2018). For all of these models, image size was defined according to the default for each model and the dataset with data augmentation was used. The parameters adopted were: batch size = 128; optimizer = Adam; dropout = 0.6 and learning rate = 0.001. Although a learning rate of 0.0001 previously demonstrated slightly better results, a higher learning rate was chosen due to time and computational costs, as well as velocity of convergence. The InceptionV3 was not tested until the end, since it was outputting poor loss values, compared with the other models. The results for these models can be found in table 1.

For the mixed input model, we combined convolutional layers for processing image inputs with dense layers for non-image inputs. For these models, we did not use data augmentation, due to implementation difficulties. The convolutional part of the model followed the same architecture as the “simple” models. The outputs of the convolutional layers were flattened before concatenation. The features stream consisted of one dense layer with 114 units and the ReLU activation function. The outputs of both streams were concatenated to form a combined feature vector, with dropout regularisation applied if specified. This combined feature vector was then fed into a dense layer with 114 neurons and the softmax activation function for classification. The compilation also had the same characteristics as the “simple” models. We explored various combinations of convolutional networks and features (dense layers). For the multi-input models, the best model obtained was the one that used the *three_partition* and *nine_partition* features. The results for this model are also presented in table 1.

After analysing all the models described here in relation to accuracy and loss, we realized that the most suitable model would be the multi-input, where we combine the images with the *three_partition* and *nine_partition* features. As in Groh et al., 2021, the strategy of integrating features into the model presents the most consistent accuracy scores on unseen validation data during training and generalization ability on independent test data. This was also the model that presented lower loss values, both for validation and test data.

Table 1. Accuracy and loss values obtained for the best "multi-parameter" and pre-trained models.

Model	Test Accuracy	Val Accuracy	Test Loss	Val Loss
"Multi-parameter"	0.1670	0.1319	4.0056	4.0222
VGG16	0.0537	0.0515	157.1986	154.5893
DenseNet121	0.0098	0.0104	21.2222	21.1201
EfficientNetB0	0.0379	0.0387	4.7318	4.7324
MobileNet	0.0370	0.0383	4.7273	4.7263
ResNet50	0.0065	0.0072	165.4557	167.1739
Multi Input	0.1425	0.1446	3.2242	3.2057

This result contradicts our initial predictions, where a greater expectation prevailed for pre-trained models, as they can transfer previously learned patterns. During our testing, these models generalized too much, and despite having increased the size of the networks, we were not able to control this phenomenon in a way that saved consistency over the epochs that the multi-input model presented to us.

4. Future work

Having had more time, several aspects would be corrected on our current approach, such as implementing code to use the F1 score as evaluation metric and extracting the images to an array format, instead of having them as images in folders. We acknowledge that this choice caused code and time constraints.

Having this settled, we could have experimented with different initial splits (e.g. 60-20-20), as well as other optimizers and activation functions. For the multi-input models, we could implement a methodology that allowed the use of data augmentation. Although it did not comprise a great number of instances, we could also develop a way to extract the missing images from *url_alphanum*.

Other interesting approaches would be when splitting the data, pay attention to the images' skin colour, to build a model more robust to ethnicity bias, or using ROC curve as evaluation metric and developing an ensemble with the model with best recall and best precision. Despite not being a conventional methodology, an autoencoder could be used for feature extraction.

5. References

Brinker, T. J., Hekler, A., Enk, A. H., Klode, J., Hauschild, A., Berking, C., ... & Schadendorf, D. (2019). Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task. *European Journal of Cancer*, 113, 47-54.

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., ... & Halpern, A. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), hosted by the International Skin Imaging Collaboration (ISIC). *arXiv preprint arXiv:1710.05006*.

Groh, M., Harris, C., Soenksen, L., Lau, F., Han, R., Kim, A., ... & Badri, O. (2021). Evaluating deep neural networks trained on clinical images in dermatology with the fitzpatrick 17k dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1820-1828). https://doi.org/10.48550/arXiv.2104.09957

Haenssle, H. A., Fink, C., Schneiderbauer, R., Toberer, F., Buhl, T., Blum, A., ... & Tschandl, P. (2018). Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Annals of Oncology*, 29(8), 1836-1842.

Harahap, N., & Zulkifli, F. (2023). Web Application Development: Skin Lesion Classification Using Transfer Learning InceptionResNet-v2. *International Journal of Electrical, Computer, and Biomedical Engineering, 1*, 65-75. https://doi.org/10.62146/ijecbe.v1i2.13

Keras. (n.d.). Keras API documentation. Retrieved from https://keras.io/api/

Koga, H., Kiyohara, Y., Uhara, H., & Tateishi, Y. (2020). Classification of clinical images using a pre-trained deep convolutional neural network. *Scientific Reports*, 10(1), 1-8.

Thohari, N. A., Afandi, & Triyono, L., & Hestiningsih, I., & Suyanto, B., & Yobioktobera, A. (2022). Performance Evaluation of Pre-Trained Convolutional Neural Network Model for Skin Disease Classification. *JUITA: Jurnal Informatika*, 10(1). https://doi.org/10.30595/juita.v10i1.12041

Tschandl, P., Rosendahl, C., & Kittler, H. (2019). The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5(1), 1-8.

Annexes

Annex 1

Table 2. Metadata for the provided csv file.

Feature name	Description
md5hash	Unique identifier for each record in the DataFrame
fitzpatrick_scale and *fitzpatrick_centaur*	Classification of skin types based on their response to sun exposure. The values range from 1 to 6, indicating different levels of skin pigmentation and susceptibility to sunburn
label	Diagnosis or condition associated with each record. It is the target variable
three_partition_label and *nine_partition_label*	The first provides a broad categorization, while the second provides more detail, specifying types of skin conditions
qc	Information about the quality of the classification of the image (e.g. wrongly labelled
url	These variables contain URLs that point to images associated with each dermatological condition
url_alphanum	The same as the previous one, the only difference being that this one doesn't contain any dots, dashes or bars, except at the end where there's always a ".jpg".

Annex 2

Table 3. Meaning of the different folders created during image extraction and data augmentation.

Folders	Meaning	Number of images
X_train	Not correctly classified training images	11156
X_train_diag	Correctly classified training images	252
X_train_final	70% of the total images (X_train + X_train_diag)	11408
X_val	15% of the total images	2505
X_test	15% of the total images	2457
X_train_aug	Images added from the X_train folder	35945
X_train_diag_aug	Images added from X_train_diag folder	3780
X_train_final_aug	X_train + X_train_aug + X_train_diag_aug	50881

Annex 3

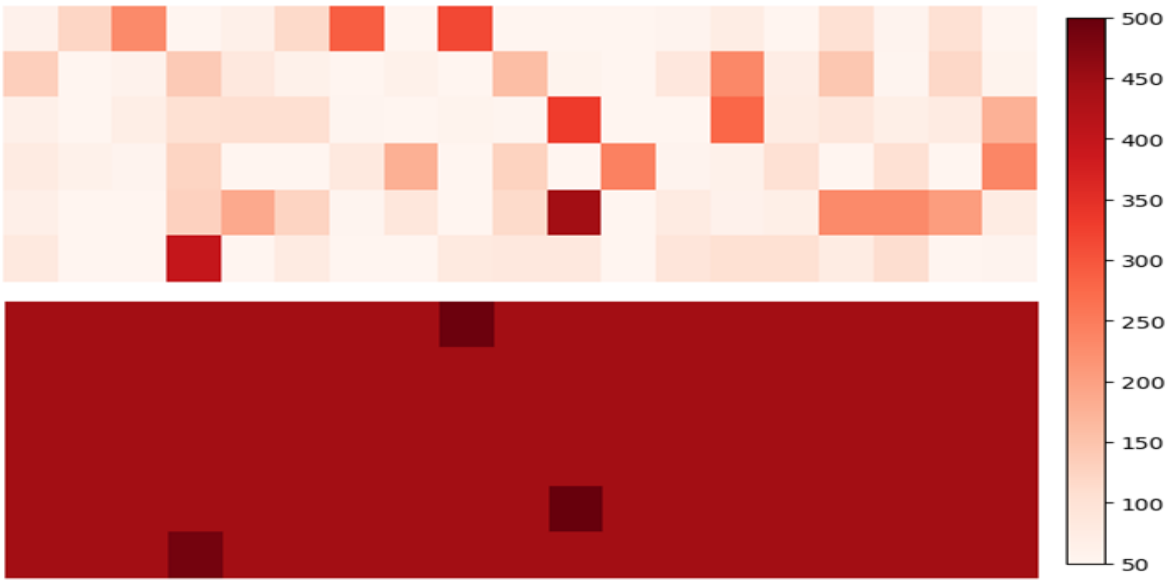

Figure 2. Schematic visualization of the quantity of images for each label before (top heatmap) and after (bottom heatmap) data augmentation.

Annex 4

Figure 3. Examples of images generated with data augmentation.