DERMATOLOGY CLASSIFICATION PROBLEM

Authors: Devora Cavaleiro, 20230974 | Jaime Simões, 20230522 | Maria Cruz, 20230760 | Carlos Rodrigues, 20230543 | Taha Ben Attia, 20230742

Required Python Libraries

These are the libraries necessary to run the code as intended:

requests; PIL ("Pillow"); os; shutil; io; urllib.parse; re; math; cv2 (OpenCV); threading; numpy; pandas; csv; matplotlib; seaborn; tensorflow; sklearn (Scikit-learn); imblearn (Imbalanced-learn); tensorboard; skimage (Scikit-image); IPython.display

Introduction

The accurate diagnosis of dermatological conditions is challenging, even for experienced doctors. In response to this need, there have been efforts to use Machine Learning, more precisely, Deep Learning models, that can improve tackling the various challenges of skin disease diagnosis.

The dataset we're using for the classification task is derived from the "FITZPATRICK17" repository. Through the development of various Convolutional Neural Network (CNN) models, and by comparing their results, we intend to contribute to the enhancement of medical image analysis. In order to achieve our best model, we used different methods. First, we preprocessed the images and metadata, filtering the misleading information, and finalized by normalizing the scales used and size of images, so the algorithm can understand the information better.

During data exploration, we discovered that the distribution of the target variable *'label'* is fairly unbalanced. Additionally, we are presented with more information about lighter skin types. Therefore, to overcome this problem, we tried data augmentation on the minority classes. This technique enables the model to better generalize across the dataset.

Finally, we applied Grid Search methods in order to find the best combinations of parameters for the models, and then we implemented a "handmade" approach, testing different combinations of convolution layers and dense layers, and as well the use of pre-trained models from Keras framework.

Analysis and Preprocessing

"FITZPATRICK17" is a dataset that combines two other dermatology datasets containing images of dermatological pathologies. Each row represents an occurrence, with an '*url*' corresponding to an image and a few columns with metadata associated with it. In total, the dataset contains 16577 rows and 9 columns, with both numerical and categorical variables in the metadata.

Our target variable is the *'label'* column, which states the skin disease diagnosis associated with each observation.

Additionally to the diagnosis, there are two types of classification for each image containing additional categories, one being '*nine_partition_label*', stating nine different categories for the diseases, and the other is '*three_partition_label*', stating if the lesion is benign, malignant or neoplastic.

‘*md5hash*’ represents a code that uniquely identifies each encounter. Since it doesn’t contain missing values, it was considered irrelevant to the outcome variable. Therefore, we decided to delete it.

The column ‘*qc*’ refers to quality control. We interpreted that it stated how reliable the labeled diagnosis is. Since 96% of the rows did not have information in this column, it could not provide predictive value for the variable outcome. Nonetheless, we filtered the incorrectly labeled rows *‘3 - Wrongly labeled’* and discarded them.

The Fitzpatrick scale is a numerical classification system ranging from I to VI that distinguishes how different skin tones react to sun exposure and their tendency to sunburn or tan (Fitzpatrick, 1988). Even being regarded as lacking the full diversity of skin types (Ware, 2020), this scale is still widely used for dermatological research.
Our initial dataset contained two distinctive columns (‘*fitzpatrick_scale*’ and ‘*fitzpatrick_centaur*’) that we assumed to be different sources (‘*fitzpatrick_centaur*’ from Centaur Labs - Medical Data Labeling Solutions), both intended to use that labeling system, as their evaluation per row differed. To address this difference, because two columns shouldn’t represent the same feature, we decided to merge them into one (*‘fitzpatrick_scale_merged’*), substituting the ’-1’ values (which we interpreted as missing values) present in the observations of one column with the label present in the other column, if it existed. Almost half (48.5%) of the missing values in ‘*fitzpatrick_scale*’ were “recovered” through this method.
There were still 291 rows that didn’t have any skin type labeling at all. Because the correlation between the two scales was now 82%, the remaining missing values were substituted by an approximation made through the method of the modes, sorted by disease category. We decided to follow this approach to avoid reducing our number of observations, as opposed to deleting them.
The initial columns were deleted, and the final values were normalized.

The *‘url’* column contains the URLs for the images. 41 rows presented missing values. As the column ‘*url_alphanum*’ doesn't have missing information, we imputed the alphanum version and manually tried to make the download accessible using the replace function based on the other URLs' punctuation and patterns. After the download, both of these columns were deleted.

Downloading images and splitting data

When downloading the images from the corresponding URLs, some did not work, because some of them did not exist or were not being recognized. This accounted for 45 instances. We decided to delete these samples since the main focus of the model should be the images and without them that extra amount of data is not beneficial for the CNN as a whole. We then encoded the categorical variables *‘three_partition_label’* and *‘nine_partition_label’* as well as the target variable *‘label’*.

Now that our data is fully pre processed, it can be split into training, validation and testing. The fractions used were 60% for training, 20% for validation and 20% for testing. One important thing to note is the use of the stratify option. Because the classes of the target variable are not equally represented, stratify makes the proportions of the classes be maintained in the three new datasets created.

Image Preprocessing

For each created dataset the images were converted into arrays with sizes (200,200,3), 200 by 200 being the size of the images and 3 the RGB filter. Different target sizes were applied for the amount of pixels being chosen and our final decision was to keep 200 by 200. A higher number will enable more precision in the training of the model (which would be considerably more computationally expensive) while making it more prone to easily overfit.

The next part of the project was dealing with the unbalanced classes. Different approaches were considered, but taking into account the dimensions of the datasets and the computational costs associated with it, we decided not to oversample all classes until they reach the amount of the most represented class. A mid term had to be reached. So first we undersampled the classes in the training dataset that have a count higher than 100 by randomly keeping only 70% of the current situations for that specific label. After that we oversampled the less represented classes (less than 50 occurrences in the training dataset) adding 90% of the initial amount for each of these labels. Now the differences between classes were a lot smaller but still notable. To resolve this we applied augmentation, setting a value that all labels need to reach as a representation (1,5% of total length of the oversampled dataset). For the images the algorithm took already existing images for each class and applied a set of random transformations to induce variability into the system while still maintaining the credibility that that class is being accurately represented. For the metadata of each row being created it simply copied the previous existing rows.

The amount of rows in the training dataset before all these transformations was 9919, and afterwards it is 14022. There was an increase and all the classes are now balanced, which makes the model able to better generalize regardless of the label.

Constructed Models

Before making the structures for the Convolutional Neural Networks, we made a function that helped us find the best combinations between some of the parameters, like the amount of filters for the convolutional layers and neurons in the dense layers. The results of this function were then applied to the handmade models. We made two separate models, one having as input only the preprocessed images and metadata and another with the data frames corresponding to the balanced classes (with augmentation). We tried two different architectures for each of the cases. There are two inputs, the preprocessed images and the metadata associated with them. They were initially trained separately and then concatenated. For the images the model has 4 convolution layers (to extract hierarchical features from the images using ‘relu’ activation function for all of them), each followed by a max pooling layer to reduce the spatial dimensions. The output of the last pooling layer is flatten in order to prepare the concatenation with the metadata section. For the metadata input the activation process was done with a dense layer of 64 neurons. The outputs of the flattened image and metadata branches were then concatenated to merge the feature representations from both branches. After the concatenation we added more dense layers before the final output layer with 114 neurons (amount of classes) with softmax activation, representing the probabilities of each class. Because it’s a multiclass problem the loss function used was ‘categorical_crossentropy’.

For the second model we took as input the augmented data sets which have the same amount of occurrences for each class label. From the two different architectures experimented, the one with less dense layers after the concatenation is giving better results in both instances.

Results: For the model without augmentation we achieved an accuracy of about 0,26 and a F1 score of 0,29, with these values stabilizing from epoch 5 onwards. With augmentation the performance was lower, with an accuracy of 0,21 and F1 score of 0,23. This could be due to the augmentation performed not being appropriate for this problem, in a sense that it creates more noise or even less variability which would hinder the model more than the benefits it brings.

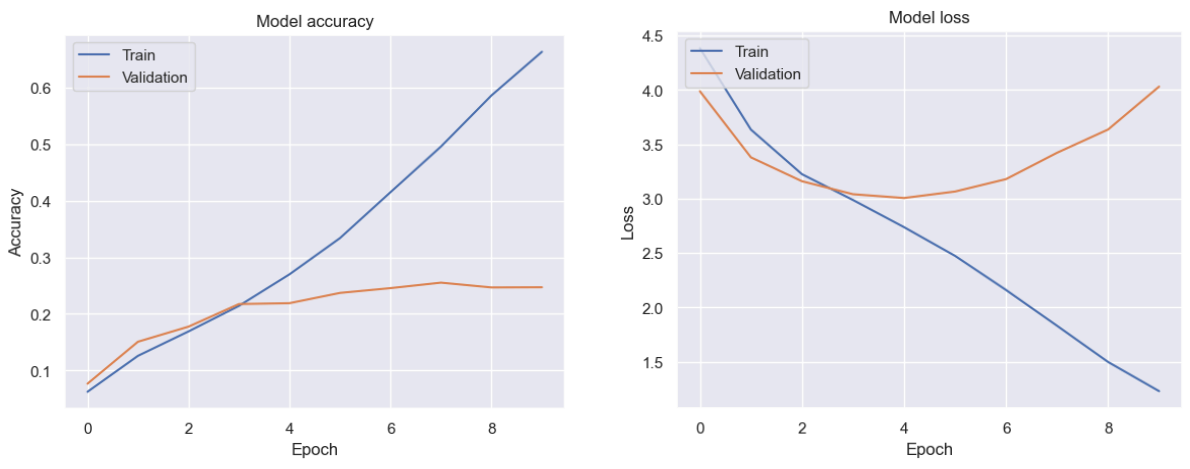

Figure 1: Accuracy and error loss for the best constructed model

Pre trained Models

To get an idea of the quality of the models built is to compare them to the pre trained ones, which have already been trained on large datasets. During our tests, we have "freezed" all layers in the pre-trained models, only considering the pre-trained weights.

To test them we performed minor pre-processing, like the work around the rows with missing url and deleting occurrences where the quality control column is *'3 - Wrongly labeled'*. The size chosen to process the images is (256,256,3), a bit higher than for built models. The models were then directly applied to the test set with the accuracy and F1-Score in the test dataset as the chosen metrics to access the quality.

Model	**Accuracy (Test)**	**F1-Score (Test)**
ResNet50	0.371	0.344
EfficientNetB5	0.369	0.351
EfficientNetB0	0.365	0.347
MobileNet	0.345	0.315
VGG16	0.330	0.300
VGG19	0.316	0.295
NASNeLarge	0.283	0.263
InceptionV3	0.259	0.243
InceptionResNetV2	0.233	0.234

Results: ResNet50 is the model with the highest accuracy (0,371) and several others performed just a little lower than this, with InceptionResNetV2 the worst performing one (0,233). These values are quite higher than the models we built, which is to be expected as they don't start the learning process from zero, they have been trained in larger and more diverse sets of data and have advanced architectures that have already been optimized.

The approach of using the pre-trained model with the metadata columns was also applied, but with worse results than the constructed models alone.

Test Set Results

To get the most accurate measurement of the quality of the models created, we made the final predictions on the test set, which represents the unseen data. For this prediction, the model that gave the best results in the validation set was used. That model corresponds to the one with no augmentation and which architecture contains less dense layers, after the concatenation of the two separate inputs. The F1 score obtained was 0.22. This value is inferior to the ones obtained in the validation set. This could be due to the model overfitting to the validation set, learning specific patterns only present in it. The size of the validation and test sets are equal but only 20% of the whole dataset, so there might be some random variability within the two. Also, the hyperparameters could be more optimized for the validation set and not perform as well in the predictions.

Future Work

The biggest challenge of this project was probably the amount of time and computational power needed to run the models. Because a big portion of our work consisted of trying different values for parameters and experimenting with different architectures, most of the time was spent in this process and we didn't explore a wider variety of possibilities for this. One example is the target size chosen for the processing of the images. It seems that the higher the value the better the models perform, but because it becomes very computationally heavy we decided to not pass a certain threshold and therefore didn't come to a limit for how much it could be increased until it starts doing more wrong than good. The result obtained of better scores for the no augmentation inputs might indicate that the augmentation being performed is not optimized. With more time, different variables and values for the augmented parameters could be experimented with to see if this trend switches. For the metadata, it should be explored the possibility of only considering some of the columns for the model input, as some of them might not have a positive impact on the learning process and this way we would also reduce the dimensionality of the problem. Lastly, we could apply transfer-learning to train some layers of the pre-trained models using our data.

References

Fitzpatrick, T. B. (1988). The Validity and Practicality of Sun-Reactive Skin Types I Through VI. *Archives of Dermatology*, 124(6), 869.

Ware, O. R., Dawson, J. E., Shinohara, M. M., & Taylor, S. C. (2020). Racial limitations of fitzpatrick skin type. *Cutis*, *105*(2), 77–80.

Centaur Labs - Medical Data Labeling Solutions. (n.d.). https://www.centaurlabs.com/