Deep Learning Project

Master Degree Program in Data Science and Advanced Analytics

Elements:

- Tomás Castilho | 20230518
- Emília Santos | 20230446
- Beatriz Santos | 20230521
- Catarina Ferreira | 20230533
- Diana Silva | 20230586

April 2024

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Index

1. Introduction

In this project, we aim to develop a deep learning model that can accurately classify 114 different skin conditions. This challenge allows us to apply the deep learning concepts learned during the trimester. We started with data exploration and preprocessing, followed by the construction and iterative refinement of multiple models to find the most effective one for dermatological diagnosis.

2. Data Pre-Processing

We started our project by thoroughly exploring and pre-processing the dataset, which is composed of 16521 images. To establish our training, validation and test sets, we allocated 80% for training and the remaining 20% split evenly for validation and test. During the download process, we encountered 52 missing files. Given the substantial size of our dataset, we opted to remove those files to maintain data integrity. Additionally, our dataset revealed an imbalance across classes so, to address that issue, we decided to do class weights technique. This technique gives more emphasis to samples from underrepresented classes, preventing bias towards the majority class and enhancing performance.

Moreover, to further enhance model robustness and generalization, we resorted to data augmentation using ImageDataGenerator from Keras Library, to improve model's robustness and avoid overfitting. Specifically, the augmentation included random rotations of up to 20 degrees, allowing the model to handle images at various angles. We also applied horizontal and vertical shifts of up to 20% of the image's width and height, respectively, which helps the model deal with off-center subjects. Additionally, shear transformations with an intensity of 20% were used to simulate a tilting effect, and zoom manipulations randomly adjusted the image scale by up to 20%. Both horizontal and vertical flips were included to ensure the model could recognize objects regardless of their orientation. We also generated three transformed images for each original image, which diversifies the dataset, thereby improving the model’s ability to learn meaningful patterns.

3. Approach

Having done the pre-processing, we were able to start developing the transfer learning with different pre-trained models, which gave us better scores in general, when compared to the handcrafted ones done afterwards. In all CNN’s that were done, there were four main steps: convolution (extraction of features from the images), implementation of non-linearity (applying ‘ReLU’ - Rectified Linear Unit - operation), pooling (reducing dimensionality) and, lastly, classification, ensured by the Fully Connected Layer, where the 'softmax' activation function is employed.

To better develop our pre-trained models, the following steps were implemented. The Base Model Layers were freezed to prevent their weights from being updated during training and, before introducing the non-linearity, we applied a batch normalization layer that normalizes its inputs, stabilizing the training, making it faster and leading to better performing models. When training the models, we decided to add the parameter class weights, as mentioned before, making the models learn more about the minority classes. For model compiling, the loss function used was 'categorical_crossentropy', which is used in the context of classification problems. Regarding finetuning, it was used in most of the pre-trained models, for instance, through the unfreezing of the base model layers. Consequently, their scores and performances slightly improved.

4. Models

4.1. Handcrafting

To craft our custom model architectures, we constructed the models using the Keras Sequential API. We based ourselves in some established models like VGG and ResNet, while also innovating with our own design architectures. Below, we present the CNN architectures for these models:

Model 1	**Model 2**	**Model 3 (VGG)**	**Model 4 (RESNET)**	**Model 5**
Conv2D (32)	Conv2D (32)	Conv2D (64)	Conv2D (64)	Conv2D (32)
Activation (ReLu)	Activation (ReLu)	Conv2D (64)	BatchNorm	BatchNorm
BatchNorm	BatchNorm	MaxPool	Activation (ReLu)	MaxPool
MaxPool	MaxPool	Conv2D (128)	MaxPool	Conv2D (64)
Conv2D (64)	Conv2D (64)	Conv2D (128)	Residual block	BatchNorm
Activation (ReLu)	Activation (ReLu)	MaxPool	Residual block	MaxPool
BatchNorm	BatchNorm	Conv2D (256)	Residual block	Conv2D (128)
MaxPool	MaxPool	Conv2D (256)	Residual block	BatchNorm
Conv2D (128)	Conv2D (128)	Conv2D (256)	GlobalAverage	MaxPool
Activation (ReLu)	Activation (ReLu)	MaxPool	Dense (softmax)	Flatten
BatchNorm	BatchNorm	Conv2D (512)		Dense (ReLu)
MaxPool	MaxPool	Conv2D (512)		Dense (ReLu)
Conv2D (256)	Conv2D (256)	Conv2D (512)		Dense (softmax)
Activation (ReLu)	Activation (ReLu)	MaxPool		
BatchNorm	BatchNorm	Conv2D (512)		
MaxPool	MaxPool	Conv2D (512)		
Conv2D (512)	Conv2D (512)	Conv2D (512)		
Activation (ReLu)	Activation (ReLu)	MaxPool		
BatchNorm	BatchNorm	Flatten		
MaxPool	MaxPool	Dense (ReLu)		
Flatten	Flatten	Dense (ReLu)		
Dense (2048, kernel regularizer)	Dense (2048)	Dense (softmax)		
Dropout (0.2)	Activation (ReLu)			
Dense (softmax)	Dense (softmax)			

Table 1 – Handcraft models architectures

The results were not satisfactory, despite conducting a random search (only in model 1), to see if it would improve. We strategically alternated between convolutional kernel sizes of 5x5 and 3x3 across the layers. This approach leverages the strengths of both kernel sizes: the 5x5 kernels are effective for capturing more global and abstract features by covering a larger receptive field per layer, while the 3x3 kernels focus on capturing finer, more localized details. Key components include a substantial dense layer with 2048 units and, in the first model, we incorporated L2 regularization and a dropout mechanism to mitigate overfitting. Model 2 was the same as 1, but without the dropout layer and l2 regularizer. We also decided to experiment the models by training them without the use of data augmentation to assess their performance. Unfortunately, this approach led to overfitting, as the models performed well on the training data, but poorly on unseen validation data. The final handcrafted models we developed were inspired by the ResNet and VGG architectures, incorporating elements like residual blocks. However, these models did not perform as expected and yielded disappointing results. Therefore, we decided to explore other options by testing pre-trained models to see if we could enhance performance.

4.2. Transfer Learning

In the transfer learning phase, we trained different pre-trained models on ImageNet, including ResNet-50, ResNet-50V2, ResNet152V2, DenseNet 121, DenseNet201, VGG16 and VGG19 and GoogLeNet, adapting it to a new task involving 114 classes. We employed a standardized set of parameters across all models to ensure consistency and comparability. We resized our images to 128 x 128 pixels instead of the standard 224 x 224 used in ImageNet training to reduce computational demands. Pre-trained layers were frozen to preserve learned features, followed by global average pooling layer to reduce feature dimensionality, and dense layer with 1024 units, which was added to enhance the learning capabilities, by providing a substantial amount of parameters. Batch normalization was applied before activation functions to unsure training stability, and we employed a learning rate of 0.01 with the SGD optimizer for efficient model optimization (Adam optimizer was explored, but SGD consistently showed better results). Given our multi-class classification task, as mentioned before, we utilized the 'softmax' activation function to probabilistically assign class labels.

4.2.1. Fine Tunning

After achieving relatively good F1 Scores following the initial training of each model, we proceeded with fine-tuning by enabling the trainable attribute for their layers. Additionally, we also adjusted the learning rate to 0.001 for fine-tuning. This process was applied to the following models: ResNet50, ResNet50v2, ResNet152V2, DenseNet121 and DenseNet201.

4.2.2. Hyperparameter Tuning

We conducted hyperparameter tuning on the two models exhibiting the highest F1 Scores on the preview phase: DenseNet201 and ResNet152V2. Leveraging the Keras Tuner library, we employed a Random Search approach to explore various configurations of optimizer choice and learning rate. The architecture takes into consideration the given optimizers such as SGD, Adam, and RMSprop and the respective learning rates. Additionally, for SGD, momentum values were also fine-tuned.

The tuning process executed over 10 trials, each one with 10 epochs, aimed to identify the configuration yielding the highest validation accuracy.

5. Evaluation

To evaluate and compare models, the metrics used were the 'AUC', 'accuracy', 'precision', 'recall' and 'F1 Score'. In the presence of class imbalance, choosing metrics like F1 Score,

Training and Validation Loss

precision, and recall is critical because they provide more meaningful insights into model performance across both minority and majority classes. Regarding the last five metrics and the 'loss' measure, some plots were built for each model, to diagnose problems in learning, such as overfitting or underfitting. The following figures show these metrics of performance for the two models that exhibited the highest F1 Scores (DenseNet201 and ResNet152V2).

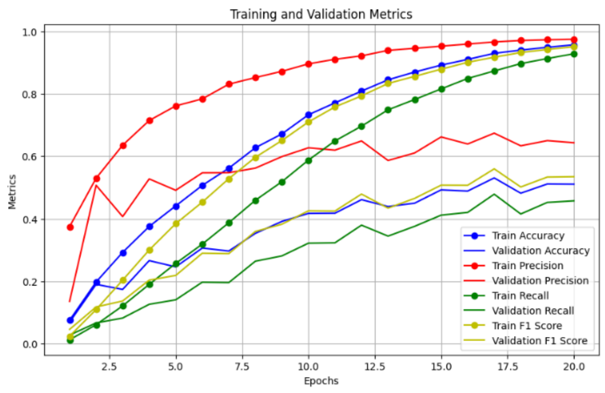

Figure 1 – DenseNet201 Model evaluation metrics

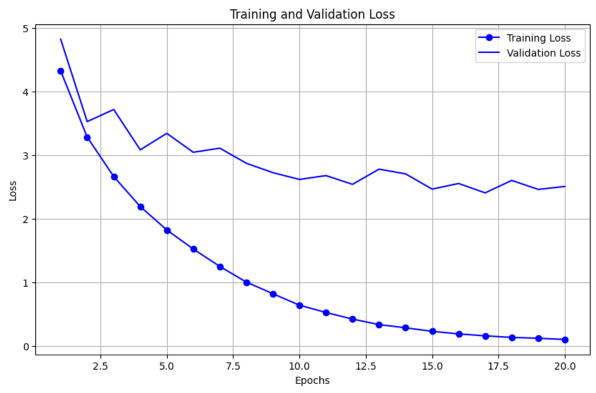

Figure 2 – DenseNet201 Model Loss

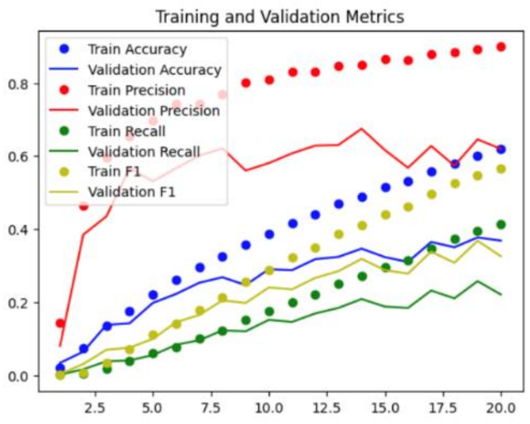

Figure 3 – ResNet152V2 Model evaluation metrics

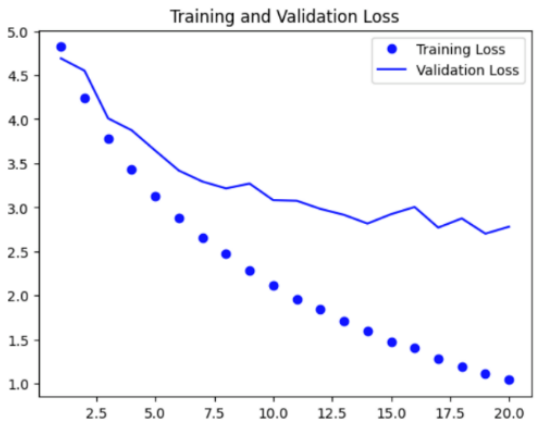

Figure 4 – ResNet152V2 Model Loss

The models DenseNet201 and ResNet152V2 produced the most promising results, given their consistent trends of increasing accuracy and decreasing loss in both training and validation, not indicating overfitting. However, a noticeable gap remains between the training and validation metrics, which might suggest room for further optimization in terms of model generalization. Additionally, the fact that precision is much higher than recall is a critical aspect in the context of medical diagnostics. While high precision means that the diagnoses the model makes are likely to be correct, the lower recall indicates that the model is missing a significant number of true cases, which could lead to under-diagnosis of conditions. The remaining models mentioned in section 4 were outperformed by these models.

6. Best Model

After conducting the random search hyperparameter tuning process, we concluded that the best model was the DenseNet201. This model was trained using an SGD optimizer with a learning rate of 0.011629533934414977 and a momentum of 0.8. Over the 20 epochs, the model demonstrated significant progress achieving a validation accuracy of 0.51 and a validation precision of 0.6434. Notably, the model showed a significant reduction in validation loss, decreasing from 4.821 to 2.507, indicating its rapid ability to learn without overfit. Moreover, the model's F1 Score reached 0.5375.

7. Limitations and Future Work

The most relevant limitations that we came across during the development of this project were, essentially, the lack of computational efficiency and the high number of classes (114 classes). Both aspects made the learning process of each model very long and, consequently, led to a running time that was very time-consuming for us. Therefore, the improvement process became difficult since to run the models once again, it would always take a very significant amount of time, even upon the usage of a GPU.

One potential improvement for our project could have been the use of the Fitzpatrick scale. This scale categorizes skin types based on their response to ultraviolet light and could have provided a structured approach to analyze our model's performance across different skin types, ensuring better generalization and relevance in dermatological applications. However, our dataset had an uneven distribution of skin types, with more common types (1, 2, 3) than less common ones (4, 5, 6). This imbalance likely affected the model's ability to accurately predict across all skin types. A more balanced dataset could have led to more equitable and effective model performance.

In terms of future work, if we had more time, we could develop different models, more trials and epochs per model, making the learning process more efficient and having better scores.

8. Conclusion

To sum up the project, which had as main objective the development of a Deep Learning model to address a dermatology classification problem, our team started by doing a brief exploration of the dataset and, afterwards, the pre-processing execution, principally, composed by the data split (train, validation, and test sets) and the data augmentation.

We created 8 models in total, in the transfer learning phase. In terms of handcraft models, 5 models were developed. The model which obtained the best performance was DenseNet201, with an F1 Score of approximately 0.538, after the fine tuning and hyperparameter tuning processes. In terms of the handcraft models, we were not so well succeeded, since the best model had an approximate F1 Score of 0.126.

Although we reached some good and suitable scores on the transfer learning, given the problem, the dataset, and the available computational resources, we believe that with some more time and better resources, the development of the project could have been more successful.

9. References

- *Team, K. (n.d.). Keras documentation: BatchNormalization layer. https://keras.io/api/layers/normalization_layers/batch_normalization/*
- *Boesch, G. (2024, March 12). VGG Very Deep Convolutional Networks (VGGNet) – What you need to know. viso.ai. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/*
- *Igareta, A. (2022, March 30). Dealing with Imbalanced Data in TensorFlow: Class Weights. Medium. https://towardsdatascience.com/dealing-with-imbalanced-data-in-tensorflow-class-weights-60f876911f99*

- *Team, K. (n.d.-b). Keras documentation: Transfer learning & fine-tuning. https://keras.io/guides/transfer_learning/*
- *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).*