NOVA IMS Information Management School

Deep Learning Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Dermatology Problem

Afonso Anjos, 20230527

Diogo Reis, 20230481

Filipe Pereira, 20230445

João Machado, 20230426

Kenza Balsam Boukhris, 20230604

April, 2024

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

Index

1. Abstract

This report details a project conducted on the application of Convolutional Neural Networks (CNNs) to classify skin conditions using the Fitzpatrick17k dataset. Therefore, deep learning methods were used to devise algorithms for more accurate dermatologic diagnosis. We benchmarked several well-known architectures, such as LeNet and AlexNet, as well as pre-trained models, including VGG16, InceptionV3, and ResNet101V2, using transfer learning for improved model performance on clinical images. Secondly, we adopted an integrated multi-input strategy to minimize bias and contribute to model generalization across diverse skin types. We chose the appropriate evaluation metrics that are able to handle the complexity of medical image classification, especially with an imbalanced dataset, which are accuracy, loss, and F1-Score. This was evidence that CNNs can significantly influence the healthcare field by detecting and classifying skin diseases for early diagnosis and possibly better outcomes of the patient. Our future work includes developing new architectures, enhancing interpretability and integrating multimodal data to make the models' diagnostic abilities robust further.

The model that returned the best results was the *Multi-Input Resnet101V2* with a dropout of 0.4 with an accuracy of 0.374, an F1-Score of 0.471 and 3.826 of loss.

2. Task Definition

Convolutional neural networks are some of the most impactful advancements in the computer vision field, these are a category of Neural Networks that have demonstrated great efficacy in areas such as image recognition and classification.

CNNs have been capable of identifying faces, objects and animals apart from providing visual capabilities to robots and powering autonomous vehicles. These models are able to accurately classify images with multiple classes by learning their features, firstly low-level features like curves and edges and then building up to complex patterns.

Many skin diseases can be fatal, like skin cancer, and early identification significantly impacts the therapy of the illness. The performance advances of CNNs suggest that these models could be a strong tool to diagnose accurately and more efficiently these types of diseases.

The objective of this project is to develop a deep-learning model to address a dermatology classification problem that performs well on unseen data. We will be using the dataset taken from the Fitzpatrick17 repository, which is a combination of two dermatology datasets, “DermaAmin” and “ATLAS Dermatologico”. The Fitzpatrick 17k dataset comprises 16,577 clinical images with labels indicating both skin condition and skin type based on the Fitzpatrick scale, which is a numerical classification system for human skin colours. More specifically the classification will be done on 114 different skin conditions, with at least 53 images and a maximum of 653 images per skin condition.

3. Evaluation Measures

The evaluation of the CNN’s is a crucial step in their development and subsequent deployment. It gives us insights into a model’s performance and allows us to measure their ability to predict new unseen data, in image classification tasks. When evaluating a CNN, we are able to know its weaknesses and thus ensure that the resulting output metrics meet our goals, whether from a generalisation standpoint or simply by achieving the desired accuracy.

In our task, we opted to keep track of both accuracy and loss, as well as the F1-Score measure. Accuracy is a common and direct metric for evaluation in classification tasks, it's a ratio between the number of correct predictions and the total number of predictions. The problem is that it's not the best indicator for imbalanced datasets and can very easily be misleading for it tends to favour the majority classes, while not taking into account the minority ones [1]. Since our data was inherently imbalanced, we needed to evaluate our computer vision models using the F1-Score, because this metric considers both precision and recall, in other words, it provides us with a balanced measure between both metrics [2]. To be able to use the F1-Score metric in our model's evaluation we had to create a custom function, so we could monitor the model performance throughout the training process. In our function we opted to follow an approach that calculates the weighted form of F1-Score which emphasizes the importance of the minority classes. [3]. Finally, the third metric we tracked, along with the other two, was the loss function which measures the quality of the prediction of a neural network model in a specific task, which represents the difference between the predicted outputs and the true targets and by minimizing its value during the backpropagation process we are able to achieve better results during training and consequently during the test evaluation [4].

Throughout our evaluation, our goal was to train the models we created using the training and the validation sets, where we monitored all three metrics, in order to track our progress and check for generalization problems. For this task, we always compared both the accuracy as well as the loss (per epoch) on the training and validation accuracy sets, and the F1-Score, also on both sets. Finally, we evaluated the models using the test set, to ensure that they were capable enough to predict images/data never seen before.

4. Approach & Results

4.1. Experimental Setup

To be able to replicate our work and validate its results, some Python libraries and frameworks need to be imported. First, Urllib for downloading data, and Pandas for managing the CSV file that contains our images dataset. Also, Numpy for data preprocessing using numerical operations. Then, TensorFlow and Keras, importing different components such as tensorflow.keras.models for sequential and functional API models, tensorflow.keras.layers for neural network layers, tensorflow.keras.optimizers for features optimization, tensorflow.keras.preprocessing.image for handling image preprocessing, as well as tensorflow.keras.callbacks to prevent overfitting and tensorflow.keras.applications to make use of transfer learning. Finally, we employed Matplotlib for data visualization during the exploration and the model training phases, OpenCV for raw image handling and Pickle for easier storage and retrieval of data. Unfortunately, since we run different models on different computers, due to incompatible versions and lack of computational power it wasn't possible to save all models that way.

4.2. Pre-processing

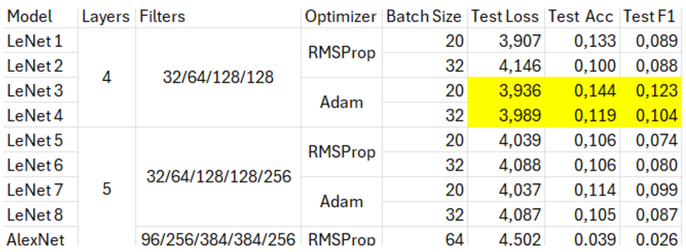
After downloading all images from the dataset, the first step was to check if all of them were downloaded correctly. Unfortunately, we found 41 missing rows and 10 links were not available, so they were all dropped. The next step was to split the dataset between training, validation and test sets. We opted to divide the data in 80% for training and 10% both for validating and testing. After that all functions - to make the plots, to evaluate the models and to calculate the weighted F1-Score were defined. Following that an image generator was created, in order to balance the data set, with the following parameters: width and height that are used to translate the images, and shear and zoom, all defined with 0.2, the rotation range that rotates the images in 40 degrees, horizontal flip that is used to invert half of the images horizontally [5], brightness that randomly adjusts the brightness of each image within a range between [0.5,1.5] [6], channel shift which is a tool to change the RGB colour channels [7] [8]. All of these data augmentation parameters were used after the rescaling between [0,1] of each colour channel.

For the multi-input models, we created functions that allowed us to map the images to their fitzpatrick skin type and label. We also created a custom function to preprocess the images that consisted of resizing the image to a standard size of 224x224 pixels and normalizing the pixel values to a range between 0 and 1 by dividing by 255. These steps made it possible for us to easily use the images along with the numerical data for modelling.

4.3. Modelling

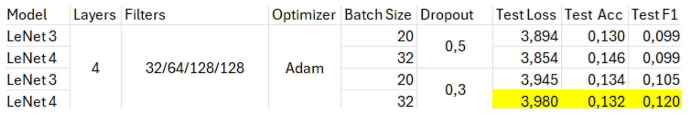
The first approach that was implemented was using the well-known *LeNet* architecture, and using the layers and filters covered in the first example taught in class. After this experimentation, the same architecture was used by adding one more layer with 256 filters. Both models were run using images with size 150x150 and tested for both different batch sizes and optimizers such as *RMSProp* and *Adam*. After that, the two models with the best results in the test set were chosen and tested with different dropout values in order to reduce the overfitting. The next step was to try the *AlexNet* architecture following the paper where it was proposed [9]. Unfortunately, this model produced the worst results. *EarlyStopping* was also implemented in all models through callbacks in order to minimize the loss while the computational power was preserved.

Overall, the best model of this section was the one with *LeNet* architecture with 32/64/128/128 filters, *Adam* as an optimizer, batch size of 32 and a dropout of 0.3 with an accuracy of 0.132, an 0.120 of F1-Score and 3.880 of loss. All results can be found in Tables [1] and [2]



Model	Layers	Filters	Optimizer	Batch Size	Test Loss	Test Acc	Test F1
LeNet 1	4	32/64/128/128	RMSProp	20	3,907	0,133	0,089
LeNet 2				32	4,146	0,100	0,088
LeNet 3			Adam	20	3,936	0,144	0,123
LeNet 4				32	3,989	0,119	0,104
LeNet 5	5	32/64/128/128/256	RMSProp	20	4,039	0,106	0,074
LeNet 6				32	4,088	0,106	0,080
LeNet 7			Adam	20	4,037	0,114	0,099
LeNet 8				32	4,087	0,105	0,087
AlexNet		96/256/384/384/256	RMSProp	64	4.502	0.039	0.026

Table[1]



Model	Layers	Filters	Optimizer	Batch Size	Dropout	Test Loss	Test Acc	Test F1
LeNet 3	4	32/64/128/128	Adam	20	0,5	3,894	0,130	0,099
LeNet 4				32		3,854	0,146	0,099
LeNet 3				20	0,3	3,945	0,134	0,105
LeNet 4				32		3,980	0,132	0,120

Table [2]

We developed multiple different pre-trained models, with the *Keras* application API, using the same dataset *Imagenet* for each one. This dataset is a large and diverse collection with tens of millions of cleaned and labelled images [10], that are used to pre-train a specific model, that by extension becomes more capable of predicting new unseen data.

The first pre-trained model used was the *VGG16 (Visual Geometry Group 16*). As the name suggests, the original model possesses 16 layers with learnable parameters, with 13 convolutional layers and 3 fully connected (dense) layers [11]. In our work, however, we used a custom version, where we removed the top layers and froze the convolutional layers of the base model so that the weights weren't updated during the training, hence preventing the model from forgetting the important features learned from the *Imagenet*. Finally, we added new layers to adapt to our task problem, a procedure called transfer learning and compiled the model using the *Adam* optimizer.

Our second and third models were the *Inception V3*, also known as *GoogleNet*, with and without dropouts. This model, unlike the previous ones, is a more sophisticated CNN, that uses multiple parallel paths, to consider different scales of image features [12]. Similarly to the previous approach, we also removed the model's top layers and freezed all the convolutional layers in the base model, while adding our own custom layers, so that we could both adapt it to our task, while preserving the model's feature extraction capabilities, trained on the *Imagenet* dataset. Though, unlike the previous attempt, this time, and for the following models, we added a custom global average pooling layer, making the feature maps easily interpreted as categories of confidence maps and avoiding overfitting on this specific layer. [13].

Lastly, our final pre-trained model consisted of *ResNet101V2* (Residual Networks 101 Version 2), with a 0.4 dropout layer [14]. The unique feature of this model is the concept of "residual connections", which allow the network to be more efficient and achieve better performance while maintaining training stability, by mitigating the problem of vanishing gradients [15]. Our version of this model followed the same line of thought as before, where we removed the top layers, freezed the convolutional layers of the base model and added the custom ones, in order to adapt to our task. The model that generated the best results was the *Resnet101V2* with an accuracy of 0.293, an F1-Score of 0.241 and a loss of 3.053. All results can be found in Table [3].

Model	Optimizer	Batch Size	Dropout	Test Loss	Test Acc	Test F1
VVG16				3,697	0,163	0,120
GoogleNet (Inception V3)	Adam	256	-	3,266	0,247	0,191
			0,1	3,257	0,226	0,177
Resnet101V2		64	0,5	3,053	0,293	0,241

Table [3]

4.4. Multi-input

For the last set of models, we took advantage of the *Keras* functional API multi-input capabilities to enhance our previous models by using the fitzpatrick_scale feature along with the images.

In the Fitzpatrick 17k dataset, there are significantly more images of light skin types than dark skin. There are 7,755 images of the lightest skin types (1 & 2), 6,089 images of the middle skin types (3 & 4), and 2,168 images of the darkest skin types (5 & 6). The rest of the dataset entries (565 images) were labeled as unknown [16].

The mean Fitzpatrick skin types across these skin condition labels ranges from 1.77 for basal cell carcinoma morpheaform to 4.25 for pityriasis rubra pilaris. Only 10 skin conditions have a mean Fitzpatrick skin type above 3.5, which is the expected mean for a balanced dataset across Fitzpatrick skin types [16].

By incorporating the skin color labels on our models, we are able to address class imbalance and enhance generalization. By leveraging both image features and skin color information, the CNN can achieve more accurate and robust classification across diverse skin types.

For all our multi-input models we decided to use EarlyStopping based on the 'val_f1_metric'. The reasoning behind this is that on our previous models the F1-Score metric was still getting higher, but the model was getting stopped due to the EarlyStopping we were using based on the 'val_loss'. We will take this approach so that we can maximize the F1-Score metric and the accuracy. This decision will help us have better overall results on our key metrics F1-Score and accuracy, on the test set. The model that returned the best results was the Resnet101V2 with a dropout of 0.4 with 0.374 of accuracy, 0.471 of F1-Score and 3.826 of loss. All results can be found at Table [4]

Model	Optimizer	Batch Size	Dropout	Test Loss	Test Acc	Test F1
			-	3,385	0,27	0,285
GoogleNet (inception V3)	Adam	256		3,305	0,261	0,248
Resnet101V2			0,4	3,826	0,374	0,471

Table [4]

5. Conclusion & Future Work

Our project demonstrated the efficiency of Convolutional Neural Networks on image classification, even in the field of dermatology. By customizing pre-trained models such as *VGG16*, *Inception V3* and *ResNet101V2* and fine-tuning them to our specific task, we managed to get satisfying results even while dealing with imbalanced datasets. Moreover, the multi-input approach with added skin type labels improved the model accuracy to a great extent and made it fairer against skin colour biases. Our experiments helped guide model performance and used accuracy, loss, and F1-Score as key indicators. We used data augmentation techniques to make our models robust to various image transformations, and early stopping mechanisms to save computational resources and prevent overfitting. The results of this research only prove that CNNs are an effective help in the accurate and timely detection of skin diseases, which are two crucial factors for the patient.

We also encountered some limitations while doing this project. We weren't able to perform cross-validation due to time constraints and computational power. This would be important to make sure our models are consistent and perform well on unseen data.

Some of our models, mainly the multi-input CNNs, required a lot of computational power, due to the sheer size of data and the complex neural networks used during training, putting to the test our CPUs and overall computer hardware. So naturally, having stronger machines would allow us to further explore new techniques and possibly achieve better results, correcting some generalization problems and attain a higher F1-score while decreasing the loss.

Another limitation we have run into was the need to have the fitzpatrick skin type data always available for our multi-input model. We might not have this information available at all times because it requires the analysis of a dermatologist, while the image information can be accessible at all times as it can be easily obtained, even by the patient himself.

Our upcoming work will focus on three main areas to make our deep-learning diagnostic tool even better. We want to first explore and try applying different and newer architectures or training techniques, such as Capsule Networks [17] and GANs (Generative Adversarial Networks) [18], to improve the performance and have higher accuracy and less loss. We are also interested in making our model results easily interpretable to make the clinician fully understand the diagnosis. Finally, making our models treat different types of data along with images, such as health records or electronic information will make our tool more comprehensive and reliable within the healthcare system.

6. References

[1] Accuracy vs. precision vs. recall in machine learning: what's the difference? (n.d.). Evidently AI. Retrieved April 25, 2024, from https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

[2] Kundu, R. (2022, December 16). F1 Score in Machine Learning: Intro & Calculation. V7 Labs. Retrieved April 25, 2024, from https://www.v7labs.com/blog/f1-score-guide

[3] How to write a custom f1 loss function with weighted average for keras? (2020, January 29). Stack Overflow. Retrieved April 26, 2024, from https://stackoverflow.com/questions/59963911/how-to-write-a-custom-f1-loss-function-with-weighted-average-for-keras

[4] Oppermann, A., & Powers, J. (n.d.). Loss Functions in Neural Networks & Deep Learning. Built In. Retrieved April 25, 2024, from https://builtin.com/machine-learning/loss-functions

[5] Chollet, F. (2016, June 5). Building powerful image classification models using very little data. The Keras Blog. Retrieved April 25, 2024, from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[6] RandomBrightness layer. (n.d.). Keras. Retrieved April 25, 2024, from https://keras.io/api/layers/preprocessing_layers/image_augmentation/random_brightness/

[7] RandomChannelShift layer. (n.d.). Keras. Retrieved April 25, 2024, from https://keras.io/api/keras_cv/layers/augmentation/random_channel_shift/

[8] Kota, D. K. (2020, May 17). Understanding Image Augmentation Using Keras(Tensorflow) | by Sai Durga Kamesh Kota | Analytics Vidhya. Medium. Retrieved April 25, 2024, from https://medium.com/analytics-vidhya/understanding-image-augmentation-using-keras-tensorflow-a6341669d9ca

[9] ImageNet Classification with Deep Convolutional Neural Networks. (n.d.).

NeurIPS Proceedings. Retrieved April 25, 2024, from https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[10] About ImageNet. (n.d.). ImageNet. Retrieved April 25, 2024, from https://www.image-net.org/about.php

[11] VGG16 and VGG19. (n.d.). Keras. Retrieved April 25, 2024, from https://keras.io/api/applications/vgg/

[12] InceptionV3 function. (n.d.). Keras. Retrieved April 25, 2024, from https://keras.io/api/applications/inceptionv3/

[13] Global Average Pooling Explained. (n.d.). Papers With Code. Retrieved April 25, 2024, from https://paperswithcode.com/method/global-average-pooling

[14] ResNet and ResNetV2. (n.d.). Keras. Retrieved April 25, 2024, from https://keras.io/api/applications/resnet/

[15] KARABULUT, M. B. (2023, December 17). Unveiling the Power of ResNet101v2: A Deep Dive into Image Classification. Mehmet Burak KARABULUT. Retrieved April 25, 2024, from https://mtburakk.medium.com/unveiling-the-power-of-resnet101v2-a-deep-dive-into-image-classification-d1a10ad02f29

[16] Groh, Matthew and Harris, Caleb and Soenksen, Luis and Lau, Felix and Han, Rachel and Kim, Aerin and Koochek, Arash and Badri, Omar. (2021). Evaluating deep neural networks trained on clinical images in dermatology with the fitzpatrick 17k dataset. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1820--1828. https://doi.org/10.48550/arXiv.2104.09957

[17] ,. (2024, March 5). , - YouTube. Retrieved April 25, 2024, from https://www.sciencedirect.com/science/article/abs/pii/S0925231222010657

[18],. (2024, March 5). , - YouTube. Retrieved April

26, 2024, from

https://www.sciencedirect.com/science/article/a

bs/pii/S0304423821007913