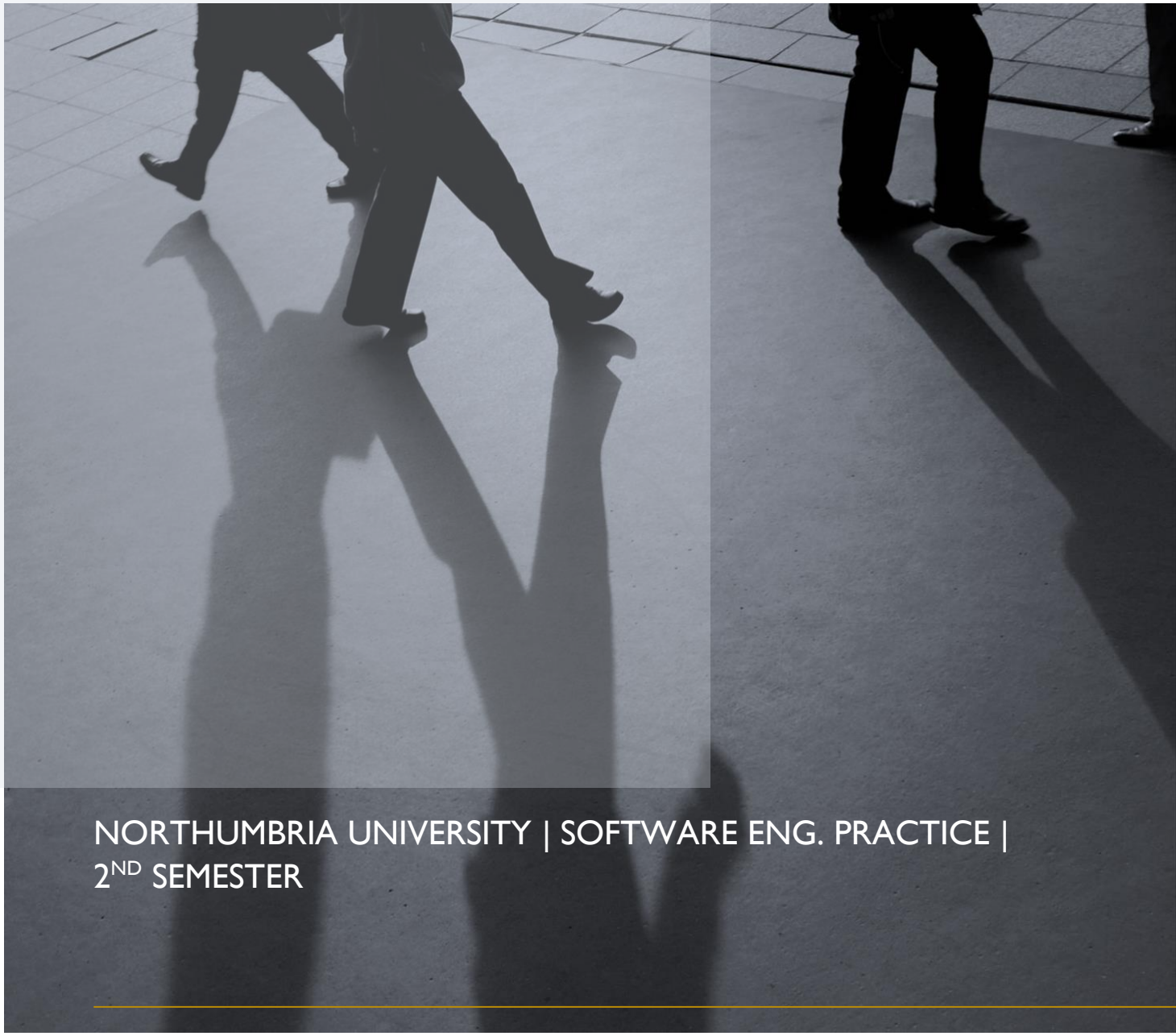
TEAM HORSE CHESTNUT

BASELINE IMPLEMENTATION



NORTHUMBRIA UNIVERSITY | SOFTWARE ENG. PRACTICE | 2ND SEMESTER

TABLE OF CONTENTS

INTRODUCTION

This document refers to the Baseline Implementation by team Horse Chestnut of the AI stream of the Software Engineering Practice module at Northumbria University. The aim of the project was to classify stages of Alzheimer given four categories: Non-Demented, Very Mild Demented, Mild Demented and Moderate Demented. For this purpose, a pipeline appropriate for Image Classification was developed, from data pre-processing to the initial implementation of a model.

DATA PREPROCESSING

The first step of this implementation was to give the opportunity to pre-process the images of the dataset. Although their format is standardised (since they result from MRI scans), it was still considered useful to experiment with some methods and see their results. As such, a function was developed to be able to apply Median or Gaussian blur to the image, alongside CLAHE or Histogram equalisation, with the possibility of not applying either blur, equalisation or even both. Regarding the blurs, which focus on reducing noise, Gaussian filtering intertwines each point in the input array with a *Gaussian kernel* and uses their sum to produce the output array, whereas Median filtering replaces each pixel with the median of its neighbours (given by a square) [1]. Histogram equalization, on the other hand, can be done either by its equally named function or by Contrast Limited Adaptive Histogram Equalisation. The first method spreads the intensity of the pixels across a wider range (a histogram) to improve the contrast of the image, fixing narrow regions with high intensities. CLAHE, on the other hand, divides the images into tiles before applying the equalisation to each of them, which prevents over-simplification of noise and preserves local information better. For the final method, no filtering method was applied but the images were subject to CLAHE [2].

The notebook also gave the option of balancing the dataset by oversampling and under sampling, made possible by copying existing images. While this approach can be beneficial to fight the uneven distribution of observations through the labels of the dataset, it is to be used with care, as it can lead to severe overfitting if no or little augmentation is used. This was eventually set to False.

Finally, the labels were encoded with One-Hot Encoder and scalers could be applied to the image to normalise its value. The main scaler was the Standard, but others were also available to be used if desired. This step, in the end, was only to be used if with NumPy arrays and was not to be considered final. After being scaled, due to the fact the images are greyscale, it was required to transform into a 3D image to be used with pretrained models. This was done by repeating the first axis.

In the end of the Pre-processing step, the images were saved into a new folder which could later be imported using *image_dataset_from_directory()* from Keras, which would also be used in the final model to divide the Train set into Training and Validation. After the importation, due to

how this function work, it was needed to certify that the images are represented with *float* values and to One-Hot encode the labels with TensorFlow.

DATA EXPLORATION

Due to the nature of the dataset, not much Exploratory Data Analysis was to be done. As such, the focus was on seeing how different the images from each category were, the distribution of the labels and the difference between the original images and their pre-processed counterparts.

The distribution of the labels in both sets are as follows:

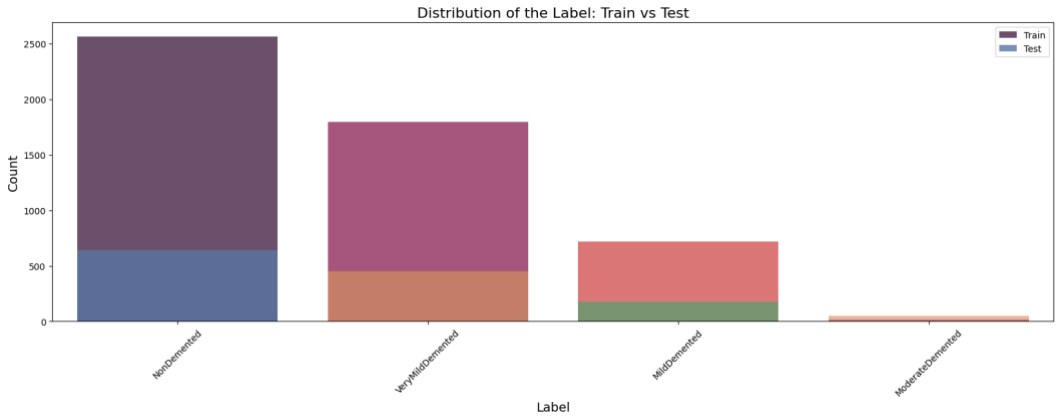


Figure 1: Distribution of the Classes

It can be seen that the labels are not evenly distributed but still follow a similar distribution in both sets. It is expectable that the model will struggle with 'ModerateDemented' and is most likely to predict 'NonDemented' the best.

MODEL IMPLEMENTATION

While this is only to be a Baseline implementation, this was the part that required the most attention. Many blocks of layers were created, including one for augmentation, to be used with the model. For this part, the *Xception* pretrained model was to be used, with some dense, batch normalisation and dropout layers being added to limit the overfit. After some runs and tests, the final model was developed, and some important metrics were evaluated, such as the F1-Score, the confusion matrix and a classification report, though these last ones did not yield the same result as the *evaluate()* method Because the Image Dataset used shuffled batches, new functions had to be implemented to ensure that the proper labels were being used to evaluate each image.

The following images display the confusion matrix for the Train (left), Validation (right) and Test set, respectively:

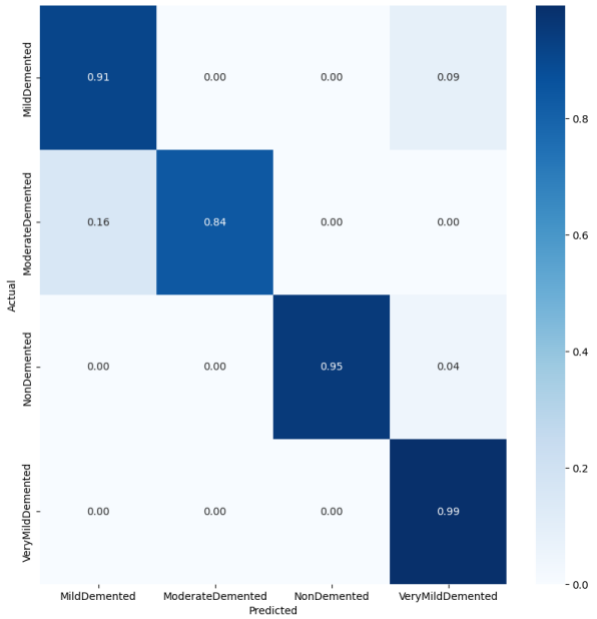


Figure 2: Train Set Confusion Matrix

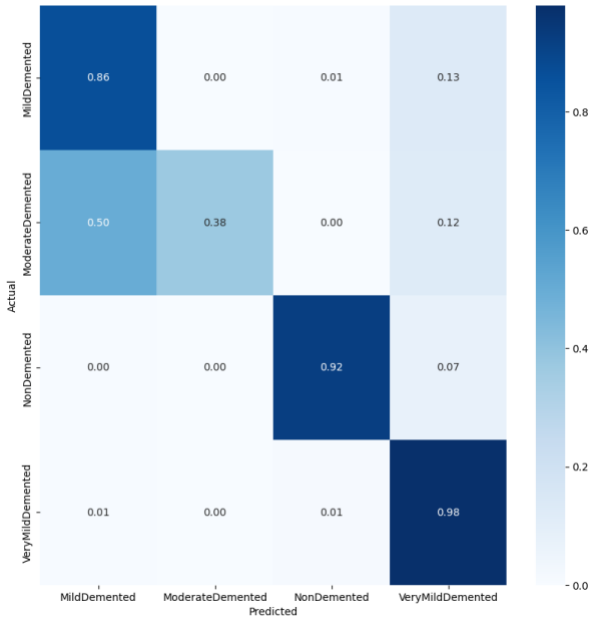


Figure 3: Validation Set Confusion Matrix

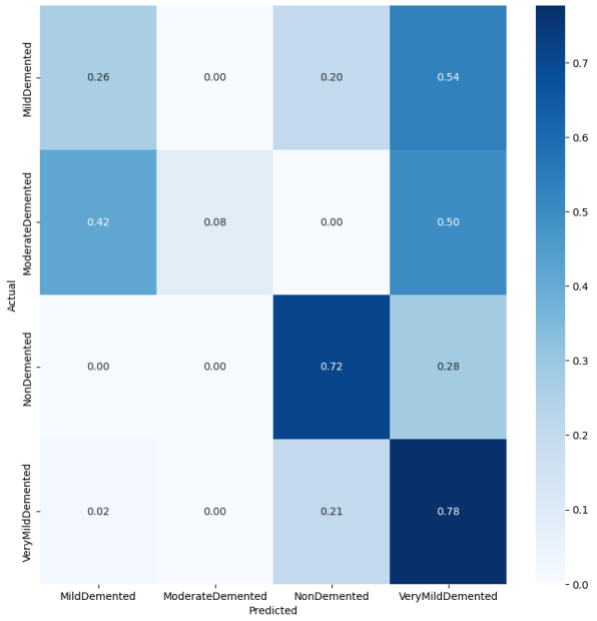


Figure 4: Test Set Confusion Matrix

Even though the initial performance of the model was, by the weighted f1-score metric, 96% on the train set, 93% on the validation set and 65% on the test set, that is only half of the truth. By analysing the confusion matrix, the model is struggles to correctly predict the 'ModerateDemented' class on the validation and test sets, also the class with the least observations. The model also tends to predict either 'NonDemented' or 'VeryMildDemented' quite more often and seems to deem 'ModerateDemented' as the same as 'MildDemented' and 'VeryMildDemented' on the test set. The large-scale misclassification of "ModerateDemented" images on the test set may suggest those images differ significantly from the ones used for training and validation, though a lesser version of this error trend can be already seen during the validation. It also started to confuse more 'MildDemented' observations with 'VeryMildDemented' ones.

FINAL CONSIDERATIONS

This Baseline allowed the team to get a better understanding of the data that is being used and pre-process it in an adequate manner. The model that resulted from it is only the first one to be considered as part of the Iterative Development that follow this part of the project. Even though every possible seed was given a set value, the performance of the model varied every run, due to how the weights start at and are computed, but it was still considered satisfactory for this part. Some experimentation was done with self-made CNNs but these did not yield similar results to the final model, with more work needed to be done in the next step.

BIBLIOGRAPHY

[1] "Histograms - 2: Histogram Equalization", OpenCV. 18 December 2015. [Online]. Available: https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html. [Accessed 9 May 2024].

[2] “Smoothing Images”, OpenCV. 8 May 2024. [Online]. Available: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html. [Accessed 9 May 2024]

ANNEX

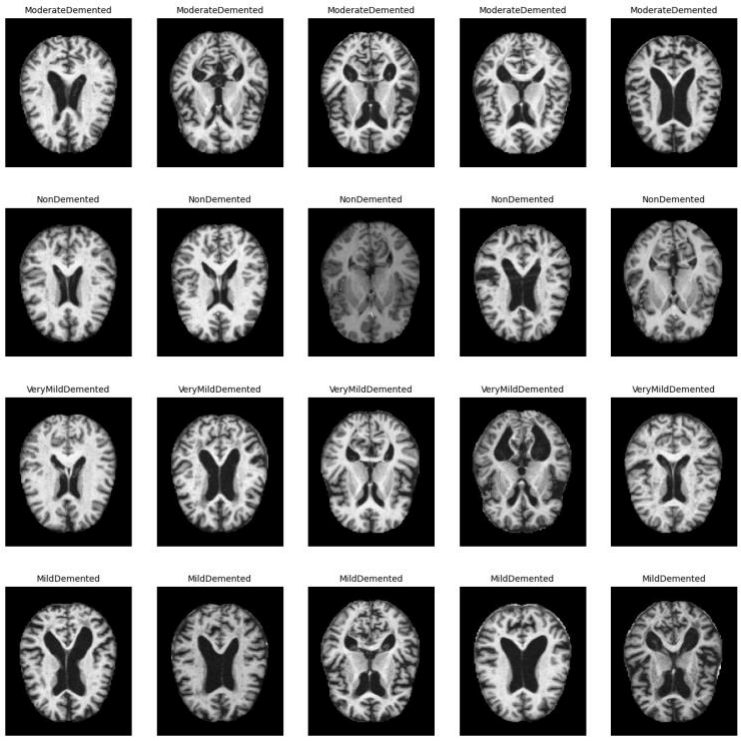


Figure 5: Example Images of each Class.

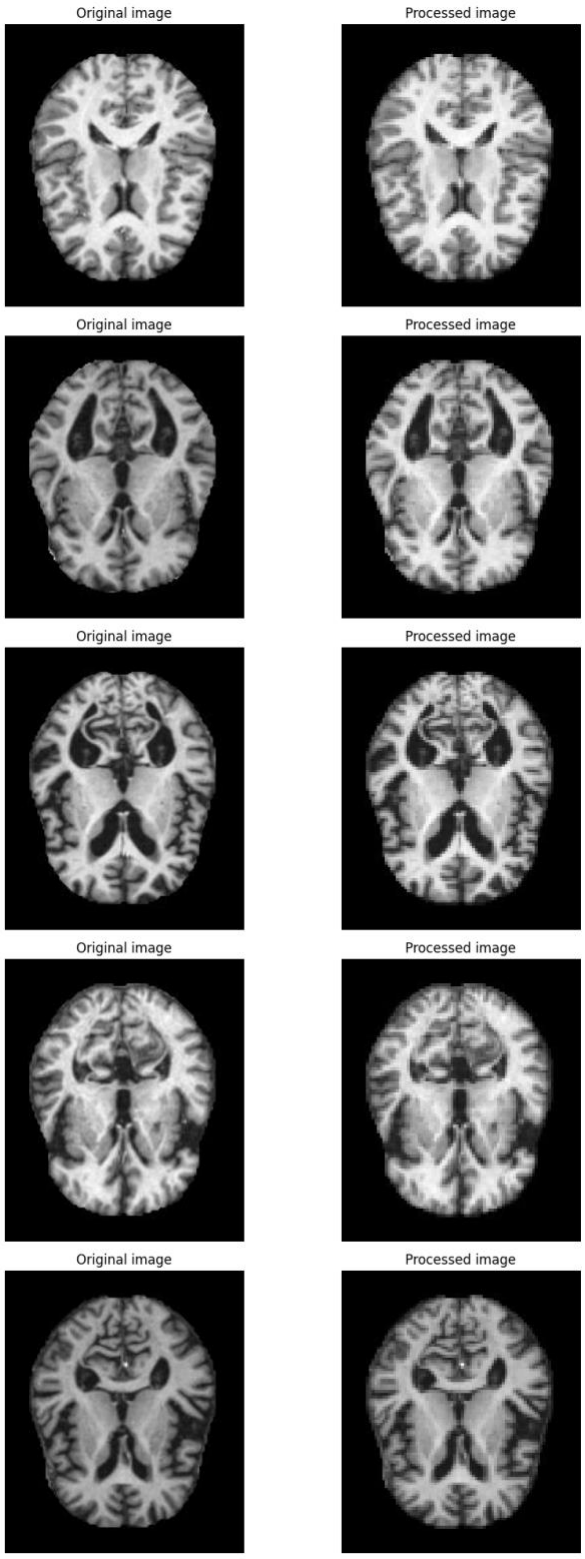


Figure 6: Comparison between the Original and Processed Images