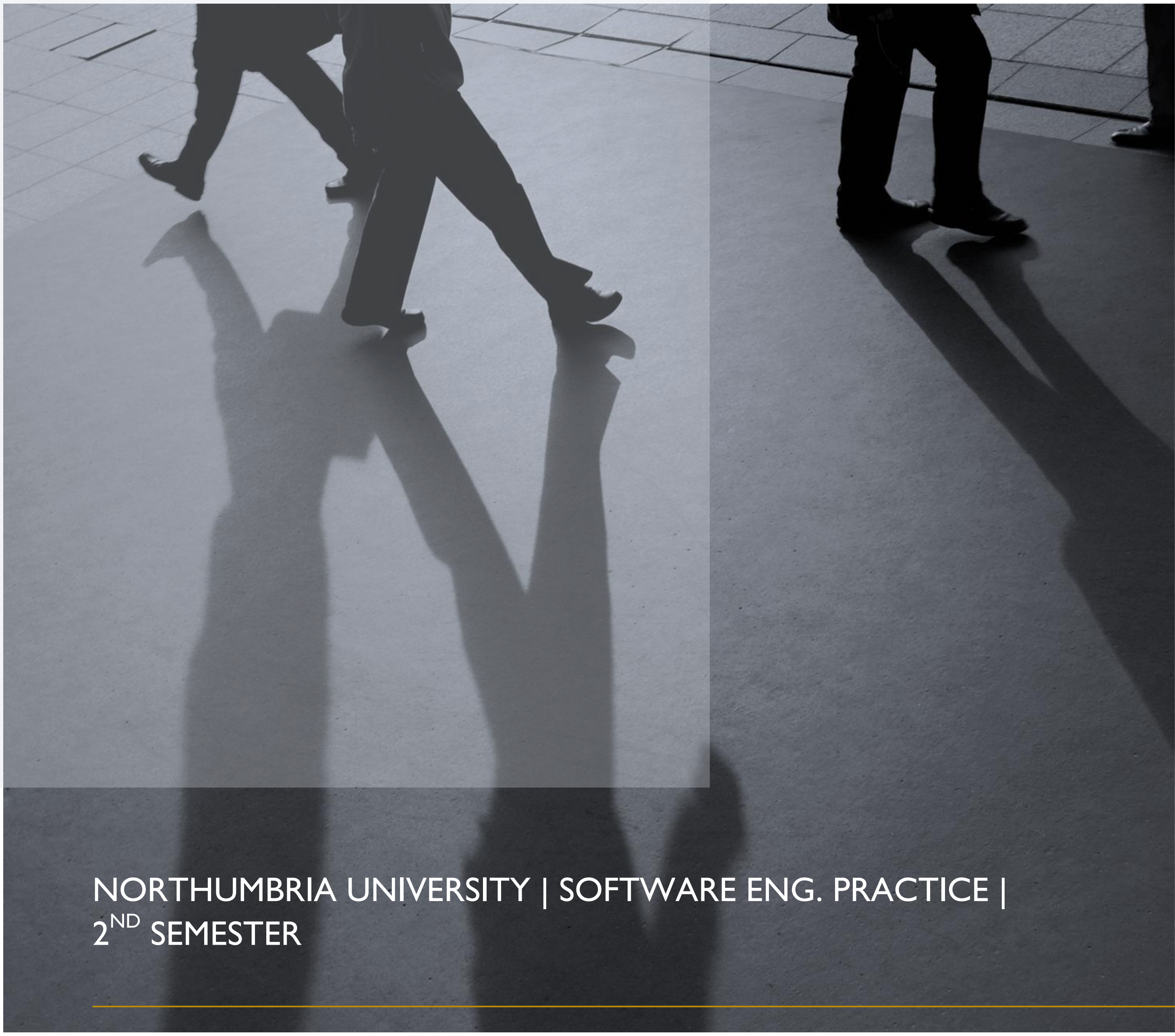
TEAM HORSE CHESTNUT

ITERATIVE DEVELOPMENT DOCUMENT



NORTHUMBRIA UNIVERSITY | SOFTWARE ENG. PRACTICE | 2ND SEMESTER

TABLE OF CONTENTS

DOCUMENT DESCRIPTION

This document will contain the elements mentioned in the brief to be a part of the Iterative Development process taken by the Horse Chestnut team to develop the final solution.

This document will contain a registration of the progress taken from the Baseline Implementation until the final solution developed, which will be demonstrated in the final demonstration. In it, will also be included the following elements:

- Pipeline Evaluations;
- Model Evaluations;
- Parameter Justifications;

In the folder where this description can be found, will also be the code, relative to the final solutions obtained after the Iterative Development process.

DEVELOPMENT PROGRESS

With the Baseline Implementation done, it was time to start the next step, Iterative Development, where multiple models were to be trained, evaluated, and compared. For this, several pre-trained models were selected and multiple CNNs were also created.

The selected pretrained models were: Xception, EfficientNet, ResNet50, VGG16, DenseNet121 and NASNetLarge. All these models were tested with a similar added structure of layers, featuring dense, batch normalisations and dropout layers, and were first tested with the parameter 'trainable' set to False, with the most promising ones lates being trained with it set to True. Finally, these models proved slow to train, and needed more epochs to converge, as the pretrained weights needed time to readjust to the new data. Still, as per the literature review, these models are some of the most encouraging ones regarding computer vision problems and have been shown to exhibit great accuracy in several distinct problems.

In terms of the CNNs that were created, these varied greatly. Multiple layer combinations were tested, such as swapping a Flatten() layer by a GlobalAveragePooling2D() one, alongside creating new blocks of layers or simply changing the order of some of the layers inside. This was thus one of the more time-consuming tasks of this step, but these models proved quick to run, which facilitated new experiments.

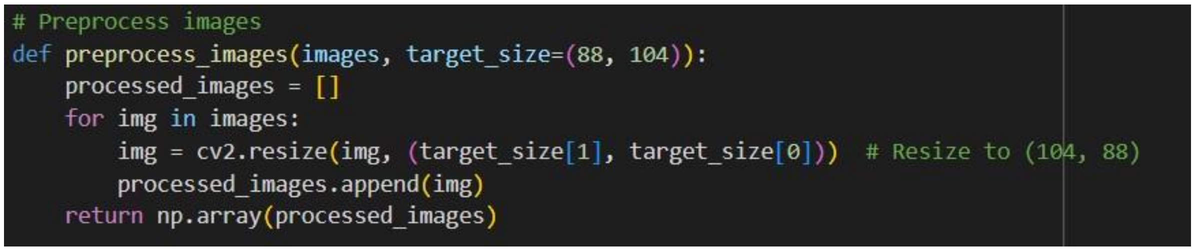
One of the problems that was faced in this step, that was already mentioned in the Baseline Development, was that the data was severely unbalanced, which made it difficult to have a reliable model, as the class 'ModerateDemented' was severely affected by it, with several models struggling to correctly predict it on the test set. To solve this, multiple methods were considered, such as SMOTE (Synthetic Minority Over-sampling Technique) and oversampling. The reason the former was discarded has to do with the fact that, to balance the classes, SMOTE creates synthetic images, which are a mix of pixels from every image of said class. While this might work well with the usual continuous data, this is not necessarily the case for images, as the algorithm might use pixels that, together, do not make sense, creating images that cannot be well interpreted by the model. Thereby, the chosen technique was the traditional oversampling and under sampling methods of copying or discarding, respectively, images from the different classes. After defining a threshold to be achieved by the classes, the dataset was now more balanced, and the model should not suffer as much. It is important to note that, as the images that were created are a copy of the original, more emphasis was to be given to the augmentation block of layers that was added do the model, which would rotate, enhance contrast or even the brightness of the images. After some testing runs and different combinations of values for these transformations, the models that were trained proved not to be considerably overfit. As a result of all of this, the pre-trained models, who suffered the most of overfit, with cases of up to 30% on some runs, had now much less overfit.

Finally, several pipelines were created, with the team trying to find new ways to solve all the issues that were presented. Using some tips provided by Professor Shanfeng, the team found a way to upgrade the overall accuracy of the different solutions that were created while being able to minimize the overfit – to a somewhat controlled margin of up to 5%.

The final proposed solutions – present in the code notebook provided with this document – include a model which was developed from scratch (a Convolutional Neural Network solution) and one which used the Xception pre-trained model as a basis for development, which proved to be te best out of all the pre-trained ones that were tested.

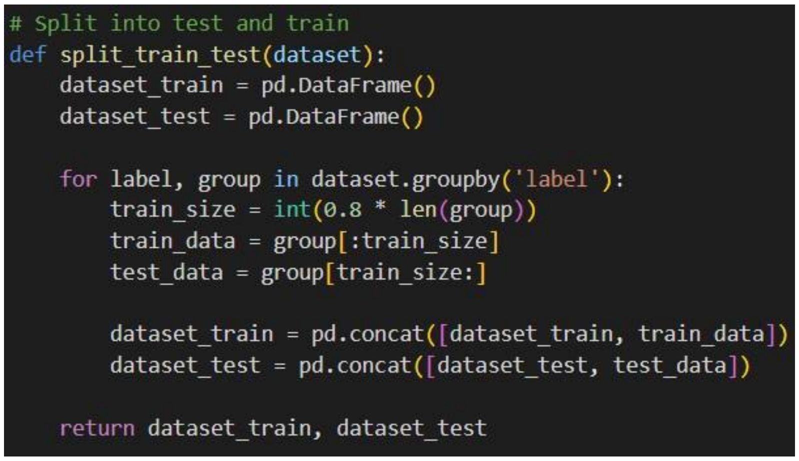
PIPELINE EVALUATIONS

For our models, we are using several pre-processing techniques to augment our images. The first augmentations we are doing is resizing the images to be 104x88 pixels wide (Figure 1). This is done so that the parameter size of the models is smaller, allowing them to be less complicated and thus less computationally expensive to train and test. It also makes the images uniform, meaning the input layer's size and the number of pixels in the image match, allowing the model to run without parameter errors.



```python
# Preprocess images
def preprocess_images(images, target_size=(88, 104)):
    processed_images = []
    for img in images:
        img = cv2.resize(img, (target_size[1], target_size[0]))  # Resize to (104, 88)
        processed_images.append(img)
    return np.array(processed_images)
```

Fig.1



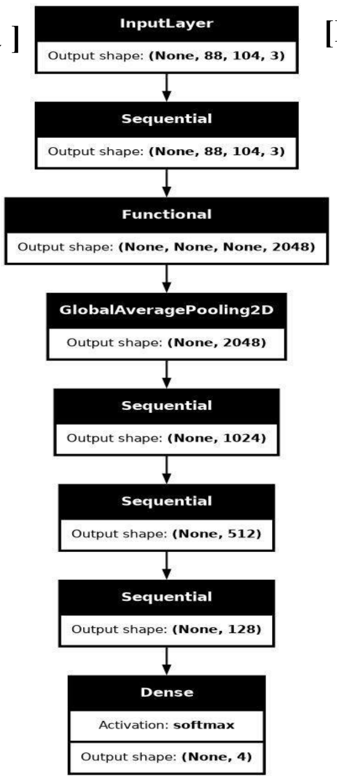
```python
# Split into test and train
def split_train_test(dataset):
    dataset_train = pd.DataFrame()
    dataset_test = pd.DataFrame()

    for label, group in dataset.groupby('label'):
        train_size = int(0.8 * len(group))
        train_data = group[:train_size]
        test_data = group[train_size:]

        dataset_train = pd.concat([dataset_train, train_data])
        dataset_test = pd.concat([dataset_test, test_data])

    return dataset_train, dataset_test
```

Fig. 2

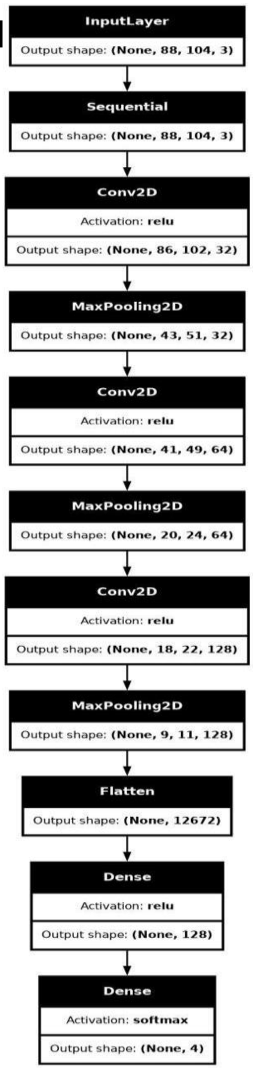
The second change we're doing is combining the test and train datasets given by the original dataset, then splitting these into new separate train and test datasets. This is because, during our initial development, we found that the models were achieving much higher accuracies in training than with testing. We suspected there may be subtle differences between the two original datasets, so we instead resampled our own at an 80% train and 20% testing split, while still making sure the proportion of each class within each was the same (As seen in Figure 2).

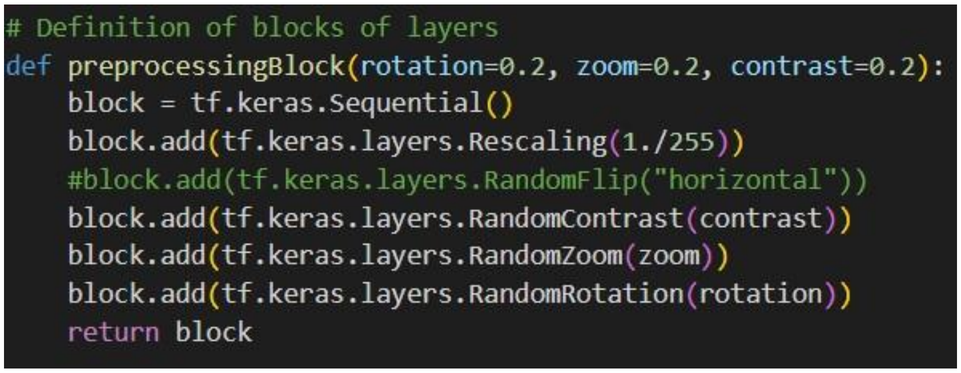
The next process was to try and balance the dataset. We struggled with this task for a longer period than expected, due to the level of imbalance in the dataset (a staggering 3300 for Non-Demented and only 64 for Moderate-Demented), something common with Alzheimer's research data (Helaly, H.A., 2022) (El-Assy, 2024). We mitigated this by both under-sampling and oversampling of the data. The over-sampling was simple to do, as we simply randomly removed parts of the dataset until we reached the target size. The smaller categories were harder to fix, as we could not simply create our own MRI images to add to the dataset. Instead, we duplicated a random selection of the categories until we reached the desired size (with a maximum oversampling rate of 4x to limit over-duplication). While this is a less-than-ideal result, as there is no new data or nuances for the models to learn from and they are not mathematically any different, we created an augmentation layer of the model to add difference.

[Figure 3a]


InputLayer
Output shape: (None, 88, 104, 3)
Sequential
Output shape: (None, 88, 104, 3)
Functional
Output shape: (None, None, None, 2048)
GlobalAveragePooling2D
Output shape: (None, 2048)
Sequential
Output shape: (None, 1024)
Sequential
Output shape: (None, 512)
Sequential
Output shape: (None, 128)
Dense
Activation: softmax
Output shape: (None, 4)

[Figure 3b]


InputLayer
Output shape: (None, 88, 104, 3)
Sequential
Output shape: (None, 88, 104, 3)
Conv2D
Activation: relu
Output shape: (None, 86, 102, 32)
MaxPooling2D
Output shape: (None, 43, 51, 32)
Conv2D
Activation: relu
Output shape: (None, 41, 49, 64)
MaxPooling2D
Output shape: (None, 20, 24, 64)
Conv2D
Activation: relu
Output shape: (None, 18, 22, 128)
MaxPooling2D
Output shape: (None, 9, 11, 128)
Flatten
Output shape: (None, 12672)
Dense
Activation: relu
Output shape: (None, 128)
Dense
Activation: softmax
Output shape: (None, 4)



```
# Definition of blocks of layers
def preprocessingBlock(rotation=0.2, zoom=0.2, contrast=0.2):
    block = tf.keras.Sequential()
    block.add(tf.keras.layers.Rescaling(1./255))
    #block.add(tf.keras.layers.RandomFlip("horizontal"))
    block.add(tf.keras.layers.RandomContrast(contrast))
    block.add(tf.keras.layers.RandomZoom(zoom))
    block.add(tf.keras.layers.RandomRotation(rotation))
    return block
```

Fig. 4

The augmentation layer of each model is an important part of the model structure and is demonstrated as the first ‘Sequential’ layer for each of the models in Figures 3a and 3b. While it does not impact the mathematical structure or core architecture of the model, it makes sure that every image in the dataset is changed randomly before being fed to the model at every epoch. This in theory should mean the model learns to differentiate non-important differences between the models (such as slight noise, image warping, or other issues related to capturing the image) and the actual differences in the brain structure of the patients. This randomness also means that the duplicate images from our oversampling are more useful, as the models get more of a chance to learn to ignore these nonrelevant differences. The changes we added were to add random contrast, random zoom, and random rotation (See Figure 4). We did not do the random flipping of the image, as this might have affected the detection of Alzheimer’s, as different effects are more prevalent in either the left or right spheres of the brain (Alzheimer's Society).

MODEL EVALUATIONS

In terms of evaluating the performance of the different models that were tested throughout the Iterative Development stage of the project, the following table was built, to represent the different performances obtained from the different models, while comparing them to the results obtained by those of other researchers:

Type of Model	**Name of Model**	**Accuracy of Alzheimer’s Classification (%)**	**Accuracy of Multiple Classification (%)**	**Sample Size**
N/A	Human (Gaugler et al., 2013)	76	37	-
Pretrained CNN (MRI)	**XbADM_Large (Ours)**	80.3	70.7	7552
Pretrained CNN (MRI)	**XbADM_Base (Ours)**	83.9	91.07	5600

CNN + SVM (MRI)	AlSaeed D, Omar SF. (2020)	85.7 to 99	N/A	417
Random Forest (Voice)	Shimoda A, et al. (2021)	86.3	N/A	1567
Pretrained ViT (Voice)	Nishikawa K, et al. (2021)	89.4	N/A	6552
CNN (MRI)	**CbADM_Large (Ours)**	89.7	80.1	7552
Ensemble Tree (MRI)	Diogo, V.S., et al. (2020)	90.6	N/A	1100
CNN (MRI)	**CbADM_Base (Ours)**	98.2	94.07	5600
Pretrained models (MRI)	Helaly, H.A., et al. (2022)	N/A	97	21816
CNN (MRI)	El-Assy, et al. (2024)	99.8	99.57	570

Figure 1 – Table of Accuracy and Sample Size

As can be analysed, our created models overperformed those presented in some papers in terms of both binary and multiclass classification, while being behind of some others. It is also worth noting that in terms of sample size, this value didn't affect much of what the performances of the models ended up being.

The table below provides another form of evaluating the accuracy of the created models, using other metrics which the group considered worthy of being used for the evaluation of the problem due to its nature:

Name of Model	**F1-Score**	**Recall (%)**	**False Positive (%)**	**False Negative (%)**
XbADM_Large (Ours)	0.708	-	4.5	6.2
CbADM_Large (Ours)	0.798	-	3.3	3.9
Diogo, V.S., et al. (2020)	-	88	1.2	0.8
Nishikawa K, et al. (2021)	0.893	-	-	-
XbADM_Base (Ours)	0.9100	92.7	0.08	0.08
CbADM_Base (Ours)	0.9106	93.1	0.07	0.08
Helaly, H.A., et al. (2022)	0.945	93.75	0.06	0.08
El-Assy, et al. (2024)	0.95	99.3	0.1	0.0

AlSaeed D, Omar SF. (2020)	0.99	96	0.31	0.15

Figure 2 – Table of F1, Recall, and False Categorising

The team chose to use these metrics to evaluate the performance of the models due to them being created to solve or provide support to a medical solution. Evaluating metrics like the False Positive and False Negative rates are important, as incorrectly diagnosing a patient can be very dangerous and potentially lead a patient to think they do not need treatment, resulting in risky consequences (i.e. the patient getting worse).

The Weighted F1-Score was also used as a metric of evaluation. Since the dataset was very imbalanced, the team thought that by using the F1-Score and adding weights to the accuracies obtained across the various categories, it made more sense as a measure of accuracy.

As can be seen from analysing Table 2, the Weighted F1-Score values obtained by our models were worse than most of those models found in various papers, which reveals that there was still room for improvement in the solutions developed by the team.

PARAMETER JUSTIFICATION

During the iterative development stage, the team's main goal was obviously to create the bets solutions possible to address the problem that was to be solved. This iterative process, as the name indicates, involved a lot of testing of different models and hyperparameters as well.

The team tried to build several solutions but, in the end, the best combinations of hyperparameters obtained after various testing runs were the following ones:

1. FOR THE PREPROCESSING BLOCK (EQUAL FOR BOTH SOLUTIONS)

- Rotation = 0.1 (Meaning 10% of the Images in the Dataset would be rotated);
- Zoom = 0.1 (Meaning 10% of the Images in the Dataset would be zoomed);
- Contrast= 0.1 (Meaning 10% of the Images in the Dataset would see their contrast be adjusted);

2. FOR THE DIFFERENT MODELS

	CNN solution	Xception + Additional Layers

Layers	Input Layer	Input Layer
	Augmentation Layer	Augmentation Layer
	Convolutional layer: 32 Filters Kernel: 3x3x3 Stride: 1x1 Activation: ReLu	Xception model
	Max-Pooling: Pool Size: 2x2	Global Average Pooling
	Convolutional Layer: 64 Filters Kernel: 3x3x3 Stride: 1x1 Activation: ReLu	Dense Layer: Neurons: 124 Activation: ReLu Dropout: 0.3
	Max-Pooling: Pool Size: 2x2	Batch Normalization
	Convolutional Layer: 128 Filters	Dense Layer: Neurons: 512

	Kernel: 3x3x3 Stride: 1x1 Activation: ReLu	Activation: ReLu Dropout: 0.2
	Max-Pooling: Pool Size: 2x2	Batch Normalization
	Flatten Layer	Dense Layer: Neurons: 128 Activation: ReLu Dropout: 0.2
	Dense Layer: Neurons: 128 Activation: ReLu	Batch Normalization
	Dense Layer(Output): Neurons: 4 Activation: Softmax	Dense Layer(Output): Neurons: 4 Activation: Softmax
Optimizer	Adam: $\beta_1 = 0.9$ $\beta_2 = 0.999$	Adam: $\beta_1 = 0.9$ $\beta_2 = 0.999$
Loss Function	Categorical Cross Entropy	Categorical Cross Entropy

The model has been set up this way, with these parameters due to a combination of empirical testing, general consensus and our own knowledge of the appropriate functions and values. We aimed to test all parameters with different values, in various combinations to discover what are the most appropriate values for each. This was set to be conducted by individually testing the model using the values said to be the best within the scientific community, then trying to change the value of one parameter until the best performing value was found. This would be repeated across all parameters. Then various combinations would be tested where 2+ parameters would be tested with their newly discovered best value. This would be an exhaustive process which would be near impossible to complete in the time frame of this assessment. As we recognized the impracticality of this plan, it was decided we would instead limit the tuning to only a few parameters. These were convolutional dropout and stride. After some testing we found that the values shown in the table above proved the best compromise of speed and accuracy.

REFERENCES

Alzheimer's Society. “Parts of the Brain.” *Alzheimer’s Society*, 2019, www.alzheimers.org.uk/about-dementia/symptoms-and-diagnosis/how-dementiaprogresses/parts-brain.

El-Assy, A M, et al. “A Novel CNN Architecture for Accurate Early Detection and

Classification of Alzheimer's Disease Using MRI Data." *Scientific Reports*, vol. 14, no. 1, 12 Feb. 2024, doi.org/10.1038%2Fs41598-024-53733-6, https://doi.org/10.1038/s41598-024-53733-6. Accessed 24 Feb. 2024.

Helaly, Hadeer A., et al. "Deep Learning Approach for Early Detection of Alzheimer's Disease." *Cognitive Computation*, 3 Nov. 2021, https://doi.org/10.1007/s12559-021-09946-2. Accessed 17 Dec. 2021.