institute of health informatics

# Graduate Programme in Health Data Science

Assessed Coursework Submission

|  |  |
| --- | --- |
| **Student candidate number:** | BSYP1 |
| **Module:** | CHME0021: Dissertation in Health Data Science  (Journal Paper) |
| **Date due:** | Monday 12th August 2019, 16.30 |
| **Word count:** (excluding references, diagrams and appendices) | 6300 |
| **Disability or other medical condition** for which UCL has granted special examination arrangements: |  |
| **Formative feedback:** | Please address in formative feedback: |
|  |
| Please ignore in formative feedback: |
|  |

# 

**Prediction of Alzheimer’s Disease (AD) from MRI using a Convolutional Neural Network**

**by**

BSYP1

A Dissertation submitted in part fulfilment of the

MSc in Health Data Science

Institute of Health Informatics

University College London

**CONTENTS**

[2](#_Toc19709034)

[ABSTRACT 5](#_Toc19709035)

[1. INTRODUCTION 6](#_Toc19709036)

[1.1 Alzheimer’s Disease 6](#_Toc19709037)

[1.2 Machine Learning approaches to diagnosis of AD 6](#_Toc19709038)

[1.3 Relevance of this study 7](#_Toc19709039)

[2. DATA and METHODS 8](#_Toc19709040)

[2.1 MIRIAD data 8](#_Toc19709041)

[2.2 Methods 9](#_Toc19709042)

[2.1.1 Convolutional Neural Networks (CNNs) 9](#_Toc19709043)

[2.1.2 Feature engineering 11](#_Toc19709044)

[2.1.3 Data management and further pre-processing 13](#_Toc19709045)

[2.1.4 CNN setup 15](#_Toc19709046)

[2.1.5 Comparing performance of CNNs 17](#_Toc19709047)

[2.1.6 Classifying subjects using majority vote 18](#_Toc19709048)

[2.1.7 Possible enhancement of CNN with text data (age, sex and MMSE) 18](#_Toc19709049)

[3. RESULTS 19](#_Toc19709050)

[3.1 Comparing different slice datasets 19](#_Toc19709051)

[3.2 Best model for best-performing slice dataset 19](#_Toc19709052)

[3.3 Majority vote results for test subjects 22](#_Toc19709053)

[3.4 Effect of adding age, sex and MMSE to the model 24](#_Toc19709054)

[4. DISCUSSION 25](#_Toc19709055)

[4.1 Comparison with other studies 25](#_Toc19709056)

[4.2 Using non-image data in the CNN 25](#_Toc19709057)

[4.3 Why does a single slice perform better than multiple slices? 26](#_Toc19709058)

[4.4 Reproducibility 26](#_Toc19709059)

[4.5 Ground-truth accuracy 26](#_Toc19709060)

[4.6 Limitations of study and possible improvements to methodology 27](#_Toc19709061)

[4.7 Conclusion 27](#_Toc19709062)

[REFERENCES 28](#_Toc19709063)

[ACKNOWLEDGEMENTS 31](#_Toc19709064)

[Appendix 1: Python code 32](#_Toc19709065)

[Appendix 2: Hardware and software used 33](#_Toc19709066)

[Appendix 3: CNN map of best model found (64, 64, 64) 34](#_Toc19709067)

**List of Figures**

[Figure 1: MIRIAD raw data – axial, sagittal, coronal and 3D surface view (own work) 8](#_Toc16496347)

[Figure 2: Example of a convolutional operation 10](#_Toc16496348)

[Figure 3: Examples of pooling operations 10](#_Toc16496349)

[Figure 4: ReLU and leaky ReLU functions 11](#_Toc16496350)

[Figure 5: A typical CNN architecture 11](#_Toc16496351)

[Figure 6: MIRIAD pre-processed data – axial, sagittal, coronal and 3D surface view (own work) 12](#_Toc16496352)

[Figure 7: Axial slices in a pre-processed scan (own work) 13](#_Toc16496353)

[Figure 8: Slice 46 for a healthy control (HC) and AD subject (own work) 14](#_Toc16496354)

[Figure 9: Data augmentation – flipping a healthy control slice across the horizontal axis (own work) 14](#_Toc16496355)

[Figure 10: The effects of weight regularisation and dropout on the validation data (own work) 17](#_Toc16496356)

[Figure 11: A typical training run showing validation accuracy stabilising (own work) 17](#_Toc16496357)

[Figure 12: Performance of different slice datasets with 3 different models (own work) 19](#_Toc16496358)

[Figure 13: ROC for tested models (own work) 22](#_Toc16496359)

**List of Tables**

[Table 1: MIRIAD demographic information 8](#_Toc16495603)

[Table 2: Balanced data split for all datasets (own work) 15](#_Toc16495604)

[Table 3: Constant values for all CNNs tested (own work) 16](#_Toc16495605)

[Table 4: Hyperparameter checking for single slice dataset – best model in bold (own work) 20](#_Toc16495606)

[Table 5: Classification of test subjects according to the best model – misclassified subject(s) in red (own work) 23](#_Toc16495607)

[Table 6: Effect of adding age, sex and MMSE data (own work) 24](#_Toc16495608)

[Table 7: Comparison with other studies (own work) 25](#_Toc16495609)

# ABSTRACT

This study builds a Convolutional Neural Network (CNN) model that classifies MRI brain scans as having Alzheimer’s Disease or not, using MIRIAD, a small database of 69 subjects (46 AD, 23 healthy controls). It also attempts to incorporate non-image data (cognitive test results, sex, age) into the CNN model.

Classification accuracy of 89% was obtained with the image data alone. Addition of cognitive test data improved the model, sex and age did not. Because this test data was used to establish the ground-truth (i.e. whether a subject has AD) the improved model is invalid since we cannot then use test data as a predictor. It does suggest, however, that non-image data can be effectively used in a CNN model.

# 1. INTRODUCTION

## 1.1 Alzheimer’s Disease

Alzheimer’s Disease (AD) – defined by ICD-10 codes F00\* and G30\* (ICD-10 Version:2016, n.d.) – is one of the most burdensome diseases worldwide (Alzheimer’s Disease International, n.d.). It “ … is associated with progressive accumulation of abnormal proteins (amyloid‑β [aβ] and hyperphosphorylated tau) in the brain, which leads to progressive synaptic, neuronal and axonal damage.” (Frisoni *et al.*, 2010 p. 67). Clinical symptoms include loss of memory, linguistic and cognitive degradation and personality and mood changes (Alzheimer’s Disease International, n.d.). The number of people with AD worldwide is estimated at 50 million in 2017, growing to 132 million by 2050, while the total cost associated with AD worldwide as of 2018 is estimated at 1 trillion dollars (Alzheimer’s Disease International, n.d.).

Although these costs and prevalence numbers appear high, they may represent a substantial underestimate of the true figures since undiagnosed AD can be as high as 80% of all cases worldwide (Alzheimer’s Disease International, n.d.). In the UK, for example, more than half of all cases are undiagnosed according to the National Audit office (cited by Getsios et al., 2012).

Although no drugs can cure AD (Alzheimer’s Disease International, n.d.), early diagnosis and treatment of AD has substantial benefits, both in terms of personal wellbeing and societal cost. A class of drugs, cholinesterase inhibitors, is effective at slowing down the progression of AD (Alzheimer’s Disease International, n.d.) and Getsios *et al.* (2012) suggest that the up-front costs of screening for AD are more than compensated for in the long run by the savings made from early treatment.

There is no specific biomarker for AD and diagnosis relies on a range of tests which include one or more of the following: cognitive assessment tests, blood tests, Computerised Tomography (CT), Magnetic Resonance Imaging (MRI), Single Photon Emission Computed Tomography (SPECT) and Positron Emission Tomography (PET) (Alzheimer’s disease, 2018). Of particular relevance to this study are MRI and the cognitive assessment test, Mini Mental State Examination (MMSE). MRI scans show atrophy of certain brain regions that are indicative of Alzheimer’s (Waldemar *et al.*, 2007), while MMSE is a quick, inexpensive test, scoring from 0-30, where higher scores are indicative of better cognitive functioning (Folstein, Folstein and McHugh, 1975). The data used in this study classifies subjects as AD if they have a MMSE score of 26 or under at baseline while a healthy control (HC) has a MMSE of 27 or above (Malone *et al.*, 2013).

Given the advantages of early-stage diagnosis of AD, any methodology that improves the speed and accuracy of AD diagnosis is important. Recently machine learning (ML) approaches that match or surpass human-level analysis of MRI have emerged (Lee *et al.*, 2017). This paper describes an attempt to create a model that uses MRI scans to classify a subject as having AD or not.

## 1.2 Machine Learning approaches to diagnosis of AD

A type of ML called supervised learning is valid in this context. Supervised learning may be thought of as the discovery of patterns in labelled data with a view to learning a model that can then be used for classification or prediction on new, unlabelled, data from a similar statistical distribution (Burkov, 2019). There are many different ML algorithms but most work on the same principle: associate the features (input variables) with the label (outcome of interest) in some type of model. Features of the data usually include numeric or text variables. However, image data is different: the features are the pixels (or voxels for 3D data), where each pixel is represented by a number.

Most ML algorithms are not as effective with image data as non-image data (Burkov, 2019). This is because images can have thousands of pixels (or voxels) and these algorithms find it difficult to cope with such a large feature space. Convolutional neural networks (CNNs), a specific type of ML, effectively solve the large feature space problem for image processing.

However, prior to CNNs, image processing was “shoe-horned” into these unsuitable non-CNN algorithms (Islam and Zhang, 2018). This required considerable manipulation of the data to transform it into a form suitable for these algorithms. Rather than pass a pixelated image to the model, for example, a histogram of pixel values would have to be created from the raw data (Chollet, 2018).

Thus, CNNs have become increasingly popular for medical image AD diagnosis. CNNs are advantageous because they can use data-driven methods for learning complex features compared to the hand-crafted features required by other ML techniques (J. Liu *et al.*, 2018) and they also have the ability to analyse data with a large feature space, such as medical images (Anwar *et al.*, 2018).

Analysis of the literature identifies two major types of CNN for AD classification from MRI:

* 3D CNNs where the whole 3D MRI is fed into a CNN capable of processing 3D images (Payan and Montana, 2015; Hosseini-Asl, Keynton and El-Baz, 2016; Backstrom *et al.*, 2018; Khvostikov *et al.*, 2018)
* 2D CNNs where 2D slices are created are created from the 3D MRI and fed into the CNN (Farooq *et al.*, 2017; Islam and Zhang, 2018; Wang *et al.*, 2018)

In this study we propose to use the latter method.

## 1.3 Relevance of this study

Although there have been several successful attempts to build a CNN to classify MRI for AD, this study uses the MIRIAD dataset. Most other papers use the ADNI (Alzheimer’s Disease Neuroimaging Initiative*,* 2017*)* or OASIS datasets (Marcus *et al.*, 2010), which have 800 and 150 subjects respectively. MIRIAD data has a total of 69 subjects and so is considerably smaller. Most papers using the MIRIAD data seem to have mainly been concerned with brain morphometry (Malone *et al.*, 2013). M. Liu *et al.* (2018) use the MIRIAD data for AD classification using a CNN, but their methods (landmark-based feature representation) are more complicated and completely different to this study. This study differs from previous studies in trying to create a CNN for AD classification with a relatively small dataset and using a relatively simple methodology. Additionally, we investigate the use of non-image data provided in the MIRIAD dataset to enhance the results of the CNN.

# 2. DATA and METHODS

## 2.1 MIRIAD data

The following description is taken from Malone *et al.* (2013, p33).

“MIRIAD (Minimal Interval Resonance Imaging in Alzheimer's Disease) is a series of longitudinal volumetric T1 MRI scans of 46 mild–moderate Alzheimer's subjects and 23 controls. It consists of 708 scans conducted by the same radiographer with the same scanner and sequences at intervals of 2, 6, 14, 26, 38 and 52 weeks, 18 and 24 months from baseline, with accompanying information on gender, age and Mini Mental State Examination (MMSE) scores.”

Demographic information is shown in Table 1Table 1.

Table 1: MIRIAD demographic information

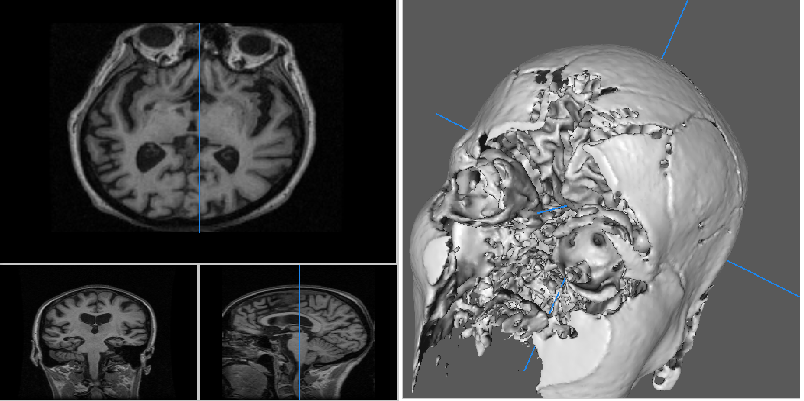
|  |  |  |
| --- | --- | --- |
|  | **Alzheimer's Disease** (N=46, Total scans = 465) | **Healthy Controls** (N=23, Total scans: 243) |
|  |  |  |
| Age at study entry (years) | 69.4±7.1 | 69.7±7.2 |
| Men | 41% | 52% |
| Mean (SD) baseline MMSE | 19.2±4 | 29.4±0.8 |

(UCL, 2018)

The data is anonymised and freely available, subject to data use conditions (UCL, 2018). Each scan is in NIFTI format (a standard file format for volumetric images) with a volumetric size of 256 x 256 x 124.

Error: Reference source not found shows axial, sagittal and coronal views as well as a 3D surface rendering of a typical raw scan.

Figure 1: MIRIAD raw data – axial, sagittal, coronal and 3D surface view (own work)



(created using Mango (Research Imaging Institute — Mango, n.d.))

## 2.2 Methods

This section describes pre-processing of raw data so it is ready for input into the CNN, as well as the CNN setup. Pre-processing of data and CNN setup are implemented using Python and its libraries, in particular Keras for the CNN, on a standalone PC (Appendix 2). A brief description of CNNs is provided in order to provide the context.

### 2.1.1 Convolutional Neural Networks (CNNs)

Neural networks (NNs) are a type of ML algorithm using successive layers to transform data into progressively more useful representations. Each layer is composed of many nodes which take their inspiration from human neurons. The nodes in each layer are connected to the nodes in the subsequent layer.

A NN is a nested function. For example, for a 3-layer NN that returns a vector, y:

y = **f**3(**f**2(**f**1(x)))

where **f**1, **f**2 and **f**3 are vector functions.

These vector functions are of the form:

**f**m(**z**) = **g**m(**W**m**z** + **b**m)

where m = layer index

**gm** = layer non-linear activation function (vector function)

**W**m (weight matrix) and **b**m (vector) = layer parameters

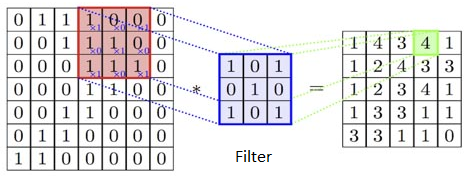
**z** = input data

(Burkov, 2019)

Each layer takes as input the output of the layer prior to it and has its own weight and bias parameters (**W**m and **b**m). The non-linear activation function (**gm**) allows non-affine, more complex, transformations of data. Batches of training data are fed into the NN. For each batch, a loss function calculates the difference between the actual and produced output. A backpropagation algorithm (optimiser) adjusts the weights in the layers such that the difference between the actual and produced output is reduced, by using gradient descent, a process of partially differentiating vector functions in the NN. This process is repeated for each batch and eventually the difference is minimised. The whole process is repeated several times for the same training data; each repetition is called an “epoch”.

CNNs are a refinement of NNs optimised for image processing. In particular, the transformations of the data involve convolution, a process that can reduce the feature space in successive layers without losing too much information. Pooling is another process that reduces the feature space. Both of these processes involve a “moving window” approach: using a filter to successively look at local parts of an image and summarising the information contained therein. The difference between the two is that convolution uses trainable filters. while pooling uses a fixed filter. In a CNN, the units in each layer are convolutional or pooling filters. Each filter in a layer exposes some characteristic of the image e.g. an edge. Figure 2 shows a typical convolution. The dot product of the filter with a “window” (shown in red) of the input array (an image, say) results in a value that indicates whether that part of the matrix represents a particular characteristic. In this case the filter is an “X” shape and the closer the “window” is to an “X” the higher the value of the dot product. Figure 2 also shows that convolution using a “moving window” can reduce the size of the input matrix (we can change this behaviour by “padding” the input with blank cells).

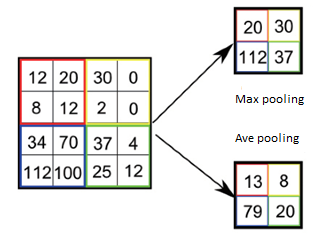
Figure 2: Example of a convolutional operation



(Chatterjee, 2017)

Different kinds of pooling operation are shown in Figure 3. For each 2x2 window in the input array, either the maximum or the average is taken. This also shows the reduction in size of the input matrix (“downsampling”).

Figure 3: Examples of pooling operations

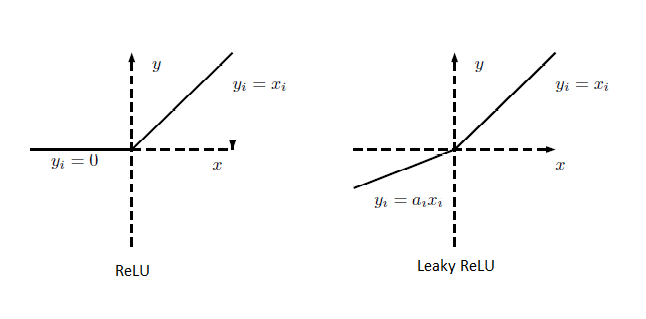


(Lee *et al.*, 2017)

Other parameters of a CNN, applied to convolutional and pooling layers, that affect the transformation process, are “padding” and “stride” which determine the change in size of the feature space.

Activation functions allow non-linear transformations of the input data. This is necessary to take advantage of the multiple layers in a CNN. Without activation functions, the hypothesis space of a layer would be limited only to linear transformations and a stack of linear layers would be redundant since adding more layers wouldn't increase the hypothesis space (Chollet, 2018). The ReLU function is one of the most popular activation functions but may “die” during the gradient descent and backpropagation process if values stick at under zero. The LeakyRelu function avoids this, since it has a slight slope ().

Figure 4: ReLU and leaky ReLU functions

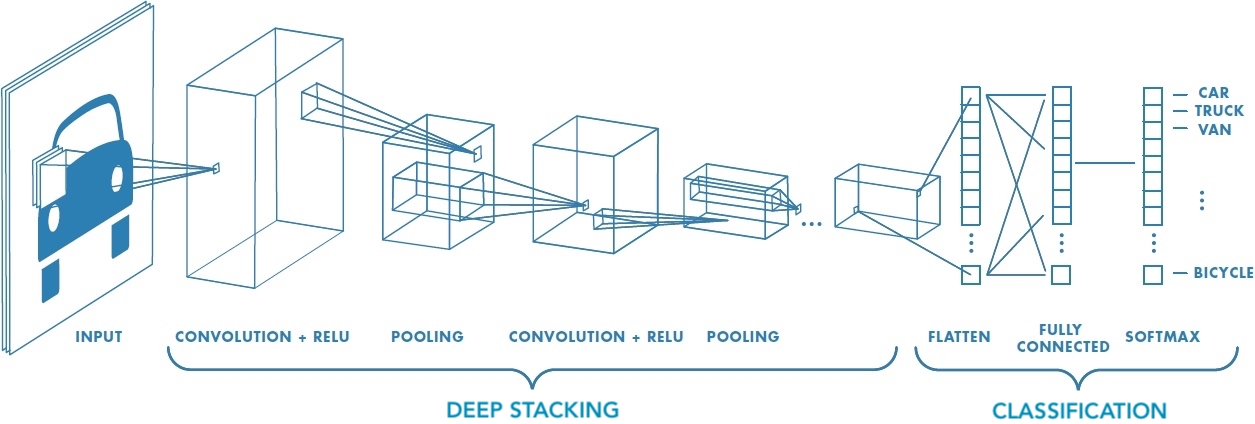


(Xu et al., 2015)

Thus, a CNN consists of several layers: convolutional, pooling and fully connected layers. Each convolutional layer consists of a certain number of trainable parametric filters. A non-linear function e.g. ReLU is usually applied to the convolutional layer. Each convolutional layer is typically followed by a pooling layer which reduces the feature space. Finally, the data is passed to one or more fully connected layers and the predicted output is produced.

Figure 5 shows a typical architecture for a CNN for binary classification. This indicates the successive reduction in size of the feature space ending up with a fully-connected layer or layers and a final softmax layer for prediction.

Figure 5: A typical CNN architecture



(MathWorks n.d.)

### 2.1.2 Feature engineering

Prior to inputting into a CNN, following the methodology of Farooq *et al.* (2017), raw MIRIAD data is pre-processed by applying spatial normalisation, bias correction and grey matter segmentation. This type of feature engineering enables the CNN to create models using fewer resources and with less data (Chollet, 2018).

Spatial normalisation is the process of mapping images from different scans onto a single template. There are two steps to this: linear transformation (e.g. translation, rotation, shear) and non-linear transformation (e.g. warping). This results in all images referencing the same coordinate space (Ashburner and Friston, 2005) and should adjust, for example, for different subject positioning when the MRI were recorded. Thus, we can compare like with like.

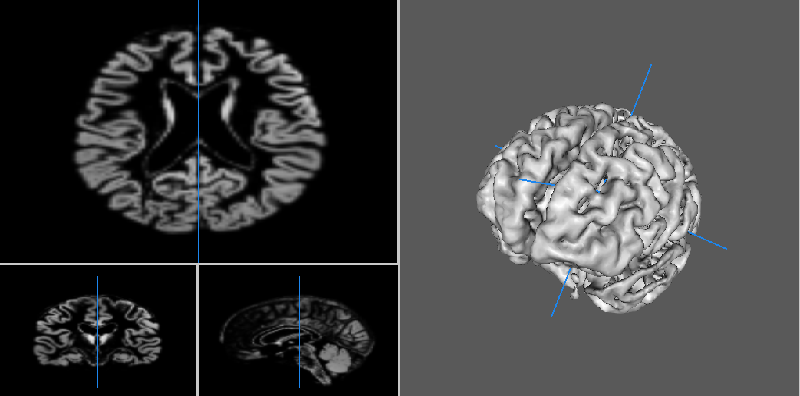
In an MRI image tissues of the same type should have the same intensity in the same image regardless of location. This never happens with MRI due to an artefact of the technology that causes smooth variation in intensity across an image (known as bias field or intensity inhomogeneity). This degrades the accuracy of many automated techniques (Song, Zheng and He, 2017). Therefore, an algorithm-based approach is required to correct for this.

Finally, we perform grey matter segmentation i.e. we extract the grey matter from the raw data. This excludes features that are unlikely to be discriminative in the classification task e.g. skull tissue (skull-stripping). Moreover, we have some evidence that grey matter volume changes may be useful in diagnosis of AD (Frisoni *et al.*, 2010). Segmentation is done in this case by using “tissue-maps”. These give the prior probability of a voxel belonging to a tissue class. Bayesian methods are then used to classify regions of the image.

The freely available SPM12 software, which is based on Matlab, is used for these pre-processing tasks (SPM12 - Statistical Parametric Mapping, n.d.). In order to aid reproducibility, the Python Nipype library interface to SPM12 is used, allowing all processing to be done in Python (Neuroimaging in Python - Pipelines and Interfaces — nipy pipeline and interfaces package, n.d.). This is a computationally intensive process that takes about 12 hours. For every raw NIFTI file, a new, smaller NIFTI file is produced of size 91 x 109 x 91.

Axial, sagittal and coronal views as well as a 3D surface rendering of a typical pre-processed scan are shown in Figure 6, which may be contrasted with Error: Reference source not found. It can be seen that many extraneous features (e.g. the skull) have been removed.

Figure 6: MIRIAD pre-processed data – axial, sagittal, coronal and 3D surface view (own work)

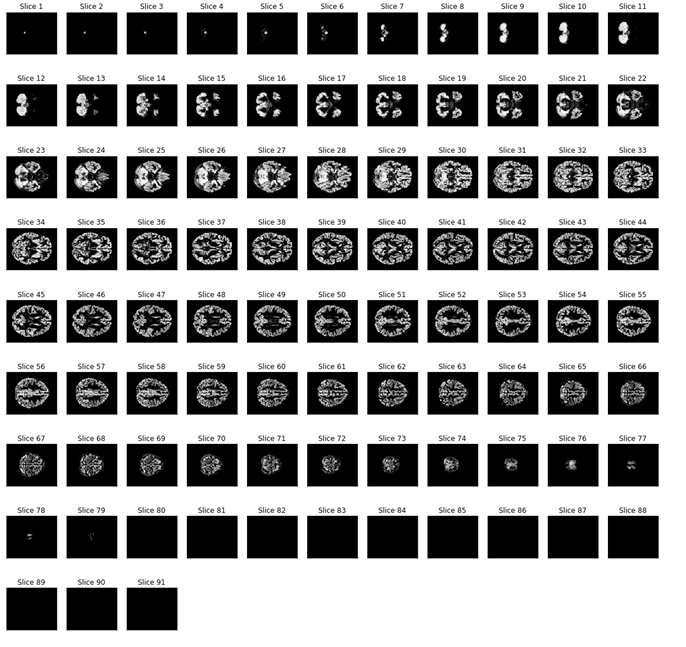


(created using Mango (Research Imaging Institute — Mango, n.d.))

### 2.1.3 Data management and further pre-processing

Following a common methodology in the literature, axial slices are programmatically taken from the pre-processed MRI scans (Farooq *et al.*, 2017; Islam and Zhang, 2018). After pre-processing there are 91 axial slices per scan. These are shown in . Slices 80 to 91 appear to be blank, but this may be due to the resolution of the image.

Figure 7: Axial slices in a pre-processed scan (own work)



Seven datasets are created, depending on the number of slices taken from each scan (1, 11, 21, 31, 41, 51 and 61 slices respectively). The single middle slice, taken for the 1-slice dataset, is no. 46. For the 11-slice dataset, additionally, 5 slices on either side of the middle slice are also included (slices 41-51), for the 21-slice dataset, 10 slices on either side of the middle slice are included (slices 36-56), etc. As Figure 7 shows, “middle” slices are likely to provide more information, as the “end” slices have progressively less data. These slices (size 91 x 109) are saved to the lossless .PNG format. Figure 8 shows slice 46 for a health control and AD subject:

Figure 8: Slice 46 for a healthy control (HC) and AD subject (own work)

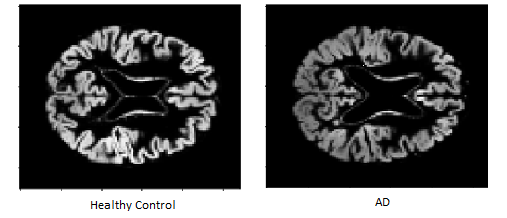
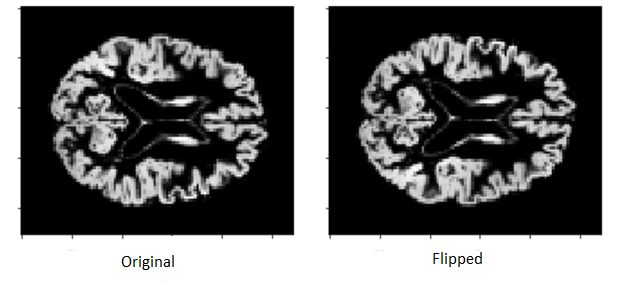


Table 1 shows that the ratio of AD subjects to healthy controls is 2:1. This is also roughly true of the number of scans (each subject has multiple scans). This leads to an imbalanced data problem whereby a model easily learns to categorise the over-represented category (AD) but not the relative scarce one (HC) (Chicco, 2017). To mitigate this, data augmentation is performed by making a “copy” of each HC slice by flipping it along the horizontal axis following the methodology of Farooq *et al.* (2017). This results in roughly the same number of slices for AD and HC subjects for all datasets. Figure 9 shows a HC slice and its augmented “copy”.

Figure 9: Data augmentation – flipping a healthy control slice across the horizontal axis (own work)



Data are split into training, validation and test datasets. This is done on a subject level. Roughly 20% of each category of subject are randomly allocated to the test dataset (10 AD subjects and 5 HC subjects). Roughly 10% of each category are randomly allocated to the validation dataset (4 AD subjects and 2 HC subjects). The reason for splitting the data on a subject level is to ensure strict separation of the test data from data used for training and validation. This is a fundamental principle in ML (Chollet, 2018; Burkov, 2019). If we split on a slice level, potentially slices from the same subject could be in both training and test datasets. Information from the test data will have been “leaked” into the training model. Although K-fold cross-validation is the gold standard in ML (Chollet, 2018), from the outset it was not used due to potentially being computationally intensive, in time terms, for CNNs (Islam and Zhang, 2018).

Table 2 shows the breakdown of number of slices for all datasets after the methodology described has been implemented:

Table 2: Balanced data split for all datasets (own work)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **No of slices per scan (Slices selected)** | **Label** | **No. of scans** | **Total slices before data augmentation** | **Total slices after data augmentation** | **Slices in training split** | **Slices in validation split** | **Slices in test split** |
|  |  | | | | | | | |
| 1 | 1  (Slice 46) | AD | 465 | 465 | **465** | 326 | 39 | 100 |
| HC | 243 | 243 | **486** | 342 | 42 | 102 |
|  |  | | | | | | | |
| 2 | 11  (Slices 41-51) | AD | 465 | 5115 | **5115** | 3586 | 1100 | 429 |
| HC | 243 | 2673 | **5346** | 3762 | 1122 | 462 |
|  |  |  |  |  |  |  |  |  |
| 3 | 21  (Slices 36-56) | AD | 465 | 9765 | **9765** | 6846 | 2100 | 819 |
| HC | 243 | 5103 | **10206** | 7182 | 2142 | 882 |
|  |  |  |  |  |  |  |  |  |
| 4 | 31  (Slices 31-61) | AD | 465 | 14415 | **14415** | 10106 | 3100 | 1209 |
| HC | 243 | 7533 | **15066** | 10602 | 3162 | 1302 |
|  |  |  |  |  |  |  |  |  |
| 5 | 41  (Slices 26-66) | AD | 465 | 19065 | **19065** | 13366 | 4100 | 1599 |
| HC | 243 | 9963 | **19926** | 14022 | 4182 | 1722 |
|  |  |  |  |  |  |  |  |  |
| 6 | 51  (Slices 21-71) | AD | 465 | 23715 | **23715** | 16626 | 5100 | 1989 |
| HC | 243 | 12393 | **24786** | 17442 | 5202 | 2142 |
|  |  |  |  |  |  |  |  |  |
| 7 | 61  (Slices 16-76) | AD | 465 | 28365 | **28365** | 19886 | 6100 | 2379 |
| HC | 243 | 14823 | **29646** | 20862 | 6222 | 2562 |

The slices in .PNG format are represented in software as an array of floating-point numbers that are fed into the CNN. Prior to inputting into the CNN, the data are normalised, putting them into the range [0,1] as recommended by Chollet (2018).

### 2.1.4 CNN setup

The number of layers and the number of convolutional filters in each layer are the two most important hyperparameters of a CNN model (Chollet, 2018). In the rest of this study we will represent a CNN configuration in the form (32, 64, 128), where the count of numbers within the brackets represents the number of layers and the numbers themselves represent the number of convolutional filters per layer. In this definition a “layer” is really a combination layer consisting of convolutional + pooling + activation units. For example, (32, 64, 128) represents a 3-layer CNN with 32 convolutional filters in the 1st layer, 64 in the 2nd and 128 in the 3rd. Similarly (32, 32, 32, 32) represents a 4-layer CNN with 32 filters in all layers.

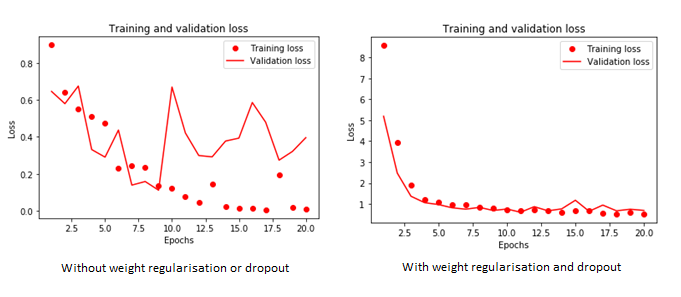
To keep the number of compared models to a manageable level, we only vary the number of layers and the number of convolutional filters per layer. All convolutional layers will have leaky ReLU activation and max pooling layers applied. All models compared will have the same loss and optimiser algorithms as well as the same weight regularisation and dropout to mitigate overfitting (explained below). There will be final flatten and dense layers (the latter with ReLU activation) and a final binary softmax layer for prediction. Table 3 shows the technical details of the constant values that were set for all CNNs tested. The default values and general architecture for the CNN setup (e.g. number of filters) were suggested by the literature, especially Sharma(2017) and Chollet (2018) and refined by experimentation.

Table 3: Constant values for all CNNs tested (own work)

|  |  |
| --- | --- |
| **Setting/Parameter** | **Value set in Keras** |
|  | |
| Loss Function | ***binary\_crossentropy***, following the advice of Chollet (2018) |
| Optimiser Function | ***RMSprop(lr=0.001***), following the advice of Chollet (2018) |
| Convolutional filter (kernel) size | ***(3, 3)*** |
| Max-pooling filter size | ***(2, 2)*** |
| Padding for all convolutional layers | ***“Same”*** (results in output feature map being same size as input) |
| Padding for max-pooling layers | ***“Same”*** (results in output feature map being half the size of input) |
| Activation function applied to all convolutional layers and final dense layer | ***Leaky ReLU (alpha=0.1)*** |
| Weight regularisation added to all models to mitigate overfitting | ***L1 = 0.001***  ***L2 = 0.002*** |
| Dropout layer added to all models to mitigate overfitting | ***0.4*** |
| Batch size | ***100*** |

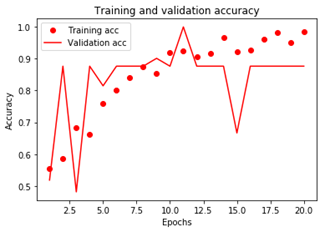
Overfitting is probably the main problem with CNNs (Burkov, 2019; Chollet, 2018). This occurs when over-complicated models are found in the training data that don’t generalise well with new data. Two ways of fighting this tendency are weight regularisation and dropout. Weight regularisation is a method of forcing a model’s weights to take on small values which leads to a simpler model, while dropout randomly drops out a percentage of output features of a layer (Chollet, 2018). The effect of these techniques is shown in . Without over-fitting mitigation techniques, over a training run, the loss value of the validation data diverges from the training data which is one indication of overfitting (Chollet, 2018).

Figure 10: The effects of weight regularisation and dropout on the validation data (own work)



Usually, training is stopped when the validation and training accuracy diverge (Chollet, 2018). At this point overfitting has started to occur. However, in this study because we generally run a model several times and take the average due to reproducibility issues (see Discussion section) we always train a model for the same number of epochs, as it’s difficult to adjust the number of epochs for each run of the model. Usually we observe that that the validation accuracy stabilises and then doesn’t worsen, so this approach seems valid (Figure 11).

Figure 11: A typical training run showing validation accuracy stabilising (own work)



### 2.1.5 Comparing performance of CNNs

When comparing slice-dataset performance (Figure 12), a relatively little-used metric, the Matthews Correlation Coefficient (MCC) is used, calculated as follows:

(TP \* TN) – (FP \* FN)

MCC = -----------------------------------------------------------------

[(TP + FP) \* (FN + TN) \* (FP + TN) \* (TP + FN)]1/2

where TP=True Positives, TN=True Negatives, FP=False Positives, FN= False Negatives

A good case is made by Chicco (2017) that the MCC metric is more balanced than metrics like accuracy and F1, because its score is high only if the classifier is good on both positive and negative predictions. The MCC is calibrated so that it ranges from -1 to +1. A value of 0 indicates a result close to chance, the closer to +1 the score is, the better the result.

### 2.1.6 Classifying subjects using majority vote

Our inputs to the various CNN models are slices. The CNN produces its prediction for each slice (HC or AD). However, there are multiple slices per subject and the possibility exists that different slices for a subject may be classified into different categories. Therefore, an algorithm is written that counts the majority prediction for the slices of each subject. In case of a tie, a random choice is made.

### 2.1.7 Possible enhancement of CNN with text data (age, sex and MMSE)

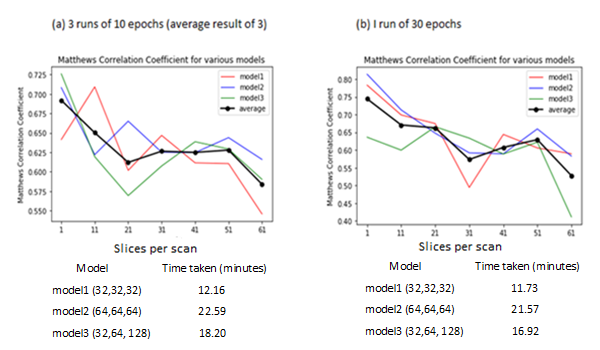
To test the effect of adding age, sex and MMSE text data to the model, normalised age and MMSE (in the range [0,1]) at the time of the scan, and sex (0.0 = Male, 1.0 = Female) are manually added to each slice array (tensor) in the top left position (we programmatically check that no image data exists in those locations). These slice arrays can be fed into any model to compare the results with or without sex, age and MMSE data.

# 3. RESULTS

## 3.1 Comparing different slice datasets

To compare the results when different slice datasets are input into the CNN, we run 3 different CNN models over each slice dataset and obtain the average. The results are shown in Figure 12 (note we use the previously described format to indicate the CNN setup, so (32, 32, 32) means a CNN with 3 layers, each of which has 32 filters).

Figure 12: Performance of different slice datasets with 3 different models (own work)



In Figure 12 (a) each model is run 3 times for each slice dataset and the MCC of the averaged results taken (the 3-run average is shown by the green, red and blue lines). The black line then averages this average over the 3 models. In Figure 12 (b) we run each model once for each slice dataset and then average over the 3 models. The reason for running each model 3 times in Figure 12 (a) is due to the problem of reproducibility (see Discussion section). It can be seen that this is a reasonably computationally intensive process in terms of time. For example, running the (64, 64, 64) model once over all slice datasets for 30 epochs each took 21.57 minutes (Figure 12 (b)).

Both graphs show roughly the same pattern: the single-slice dataset performs best, the 61-slice dataset worst, with other datasets in between. All slice datasets show a performance that is significant according to the MCC, which is greater than zero (i.e. better than chance). Different models seem to perform differently for different slice datasets.

We might have expected that more data fed into the CNN would result in better results (at least up to a certain threshold), so the fact that the single-slice dataset gives best results is surprising (see Discussion section).

## 3.2 Best model for best-performing slice dataset

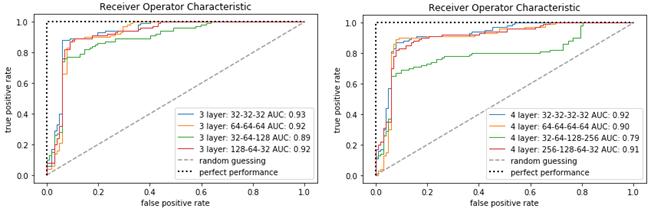
Having identified the slice dataset that gives the best result (single-slice), we explore for that dataset the hyperparameters that give the best results in terms of several metrics. As mentioned earlier we limit the hyperparameters to number of layers and filters per layer. Furthermore, following examples in the literature, we limit the number of layers to 3 or 4 which is usually adequate for good results (Chollet, 2018). summarises the results of our experiments. Note that for each model tested we take the average over 5 runs of 20 epochs each, due to problems with reproducibility (note that we don’t average the metrics overs 5 runs, rather we average the predicted probabilities and take the metrics for that average).

Table 4: Hyperparameter checking for single slice dataset – best model in bold (own work)

|  |  |  |  |
| --- | --- | --- | --- |
| **Model hyperparameters** | **Timing (minutes)** | **Metrics** | **Confusion matrix** |
| **3-layer models** | | | |
| (32, 32, 32) | 0.24 | MCC: 0.76  Accuracy: 0.88  Precision: 0.87  Specificity: 0.87  Sensitivity (recall): 0.89  F1: 0.88 |  |
| **(64, 64, 64)** | **0.38** | **MCC: 0.77**  **Accuracy: 0.89**  **Precision: 0.89**  **Specificity: 0.89**  **Sensitivity (recall): 0.88**  **F1: 0.88** |  |
| (32, 64, 128) | 0.32 | MCC: 0.59  Accuracy: 0.79  Precision: 0.74  Specificity: 0.69  Sensitivity (recall): 0.89  F1: 0.81 |  |
| (128, 64, 32) | 0.61 | MCC: 0.72  Accuracy: 0.86  Precision: 0.84  Specificity: 0.83  Sensitivity (recall): 0.89  F1: 0.86 |  |
| **4-layer models** | | | |
| (32, 32, 32, 32) | 0.28 | MCC: 0.74  Accuracy: 0.87  Precision: 0.93  Specificity: 0.94  Sensitivity (recall): 0.79  F1: 0.85 |  |
| (64, 64, 64, 64) | 0.44 | MCC: 0.78  Accuracy: 0.89  Precision: 0.92  Specificity: 0.93  Sensitivity (recall): 0.84  F1: 0.88 |  |
| (32, 64, 128, 256) | 0.44 | MCC: 0.61  Accuracy: 0.80  Precision: 0.89  Specificity: 0.92  Sensitivity (recall): 0.67  F1: 0.77 |  |
| (256, 128, 64, 32) | 1.28 | MCC: 0.74  Accuracy: 0.87  Precision: 0.90  Specificity: 0.91  Sensitivity (recall): 0.83  F1: 0.86 |  |

shows the ROC curves for the 8 models above split by 3-layer and 4-layer models.

Figure 13: ROC for tested models (own work)



From and we can see that all models have decent performance. The worst performing models seem to be the ones where the number of filters goes up from layer to layer, regardless of number of layers i.e. (32, 64, 128) and (32, 64, 128, 256). 4-layer models seem to slightly underperform 3-layer models. The “best” model of the 8 tested seems to be either the 3-layer (64, 64, 64) model or the 4-layer (64, 64, 64, 64) model which have the lowest number of misclassified instances (also reflected in their MCC scores). Although these models are very similar, the (64, 64, 64) model is preferred on balance (fewer false negatives).

Table 4 also shows how fast the models tested run on a single-slice dataset (using a GPU). Even with 5 runs (20 epochs each) of each model, most complete in under a minute – the exception being the 4-layer model (256, 128, 64, 32) which takes just over a minute.

The best model of the eight (64, 64, 64) has a (slice) accuracy of 0.89 which is reasonable. Full details of this model can be obtained from Table 3 and Appendix 3.

## 3.3 Majority vote results for test subjects

For the 15 randomly chosen subjects that were assigned to the test dataset, for the single slice dataset there are a total of 202 slices (reflecting multiple visits per subject and for non-AD subjects, data augmentation through “flipping”, see Table 2).

gives the breakdown for these subjects showing their “ground truth” i.e. whether they are labelled as having AD or not, what the best model chosen (64,64,64) predicts overall, and a breakdown of how the subject’s individual slices were classified.

Table 5: Classification of test subjects according to the best model – misclassified subject(s) in red (own work)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Subject No.** | **Ground-truth for subject** | **Category predicted by model (64, 64, 64)** | **Slices predicted as not AD** | **Slices predicted as AD** | **Total slices for subject** |
| 190 | AD | AD | 0 | 8 | 8 |
| 195 | AD | AD | 0 | 12 | 12 |
| 204 | AD | AD | 0 | 12 | 12 |
| 214 | AD | AD | 0 | 11 | 11 |
| 222 | AD | AD | 0 | 12 | 12 |
| 230 | HC | HC | 19 | 5 | 24 |
| 231 | HC | HC | 24 | 0 | 24 |
| 233 | HC | HC | 22 | 0 | 22 |
| 234 | AD | AD | 1 | 10 | 11 |
| 236 | AD | AD | 0 | 8 | 8 |
| 246 | AD | HC | 10 | 0 | 10 |
| 249 | HC | HC | 20 | 0 | 20 |
| 251 | HC | HC | 6 | 6 | 12 |
| 256 | AD | AD | 0 | 4 | 4 |
| 257 | AD | AD | 1 | 11 | 12 |

shows that only one subject, 246, in the test set is misclassified. This was a unanimous decision on all their slices. Subject 251, on the other hand, has half their slices classified as AD and half as HC. In this case of a tie, the program decides a class at random and by chance has picked the correct classification.

Subjects 230, 234, and 257 have the correct classification but not unanimously. All other subjects have a correct unanimous classification.

Digging deeper into the data for subjects 246 and 251 the raw MMSE scores as recorded at subsequent visits are as follows, in order of visit:

246: 26, 26, 26, 26, 26, 26, 28, 28, 28, 24

251: 27, 27, 27, 27, 27, 24 (this subject has ground-truth HC so has duplicate “flipped” slices, hence 12 records)

These subjects seem to be on the boundary between AD and HC since AD subjects are presumed to have an MMSE score at baseline of less than 27, while HC subjects are 27 and above (Malone *et al.*, 2013). There may be a possibility that the ground truth for these subjects is incorrect (see Discussion section).

## 3.4 Effect of adding age, sex and MMSE to the model

Table 6 compares the effect of adding (1) age and sex only and (2) age, sex and MMSE into the slice arrays as described in section 2.1.7. When we re-run the best model (64, 64, 64) with just age and sex there isn’t much change in the results compared with not adding the text data. However, when age, sex **and MMSE** are added, the result is the best model produced so far. However, as MMSE data is used to define the ground truth, it is not valid to use this as a predictor variable (see Discussion section).

Table 6: Effect of adding age, sex and MMSE data (own work)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **MCC** | **Accuracy** | **Precision** | **Specificity** | **Sensitivity** | **F1** | **Confusion matrix** |
| **Best model (64, 64, 64) without sex/age data** | **0.77** | **0.89** | **0.89** | **0.89** | **0.88** | **0.89** |  |
| Best model (64, 64, 64) with sex/age data | 0.78 | 0.89 | 0.88 | 0.87 | 0.91 | 0.89 |  |
| Best model (64, 64, 64) with sex/age/MMSE data\* | 0.83 | 0.92 | 0.92 | 0.92 | 0.91 | 0.91 |  |

**\*Although this model has very good metrics, we can’t use it since MMSE was used to define the ground truth.**

# 4. DISCUSSION

We had two aims in this study: to create an effective CNN model for the prediction of AD from MIRIAD data and to test if non-image data enhanced the results of the CNN. Both of these goals were broadly achieved. Various issues arose in the study which are explored here.

## 4.1 Comparison with other studies

Comparison with other studies is difficult due to variability in methodology, CNN architectures, data sources, and metrics reported. Nevertheless, Table 7 attempts to give some idea of study differences. Where metrics are not reported they are left blank.

Table 7: Comparison with other studies (own work)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Researcher(s)** | **Data Source (no. of subjects)** | **AUC** | **MCC** | **Accuracy** | **Precision** | **Specificity** | **Sensitivity** | **F1** |
| ***Best model in***  ***current study***  ***(64, 64, 64)*** | ***MIRIAD***  ***(69)*** | ***0.92*** | ***0.77*** | ***0.89*** | ***0.89*** | ***0.89*** | ***0.88*** | ***0.89*** |
|  |  |  |  |  |  |  |  |  |
| Liu, M. et al. (2018) | MIRIAD  (69)  Trained on ADNI | 0.97 | 0.84 | 0.92 |  | 0.91 | 0.93 | 0.95 |
| Farooq et al. (2017) | ADNI  (149) |  |  | 0.99 |  |  |  |  |
| Islam and Zhang (2018) | OASIS  (Unknown) |  |  | 0.93 | 0.94 |  | 0.93 | 0.92 |
| Wang et al. (2018) | OASIS + other  (198) |  |  | 0.98 |  | 0.97 | 0.98 |  |
| Backstrom et al. (2018) | ADNI  (340) |  |  | 0.99 |  |  |  |  |
| Hosseini-Asl et al. (2016) | ADNI  (210) |  |  | 0.98 |  |  |  |  |
| Khvostikov et al. (2018) | ADNI  (214) |  |  | 0.97 |  | 0.98 | 0.96 |  |
| Payan and Montana (2015) | ADNI  (755) |  |  | 0.95 |  |  |  |  |

Of papers reviewed, only M. Liu *et al.* (2018) use MIRIAD data and thus provide the best source of comparative results. Table 7 shows that the best model in the current study has slightly worse scores in all metrics, but nevertheless is reasonably effective. Other studies tend to report only accuracy, although this can be misleading, especially for unbalanced data (Chicco, 2017). Almost all other studies used the ADNI (Alzheimer’s Disease Neuroimaging Initiative, 2017*)* and OASIS (Marcus *et al.*, 2010) datasets which have many more subjects than MIRIAD and it’s possible that this accounts for the superior results.

## 4.2 Using non-image data in the CNN

Normalised age, sex and MMSE data were inserted into the top-left part of the image array (these cells are empty i.e. don’t contain image data) and fed into the best model found. As shown in Table 6, age and sex alone didn’t make much difference. This is probably because there is no great variation in age (in particular) or sex between AD and HC groups (Table 1). To confirm that age and sex didn’t make much difference a non-CNN model (k nearest-neighbours) was created to predict AD from age and sex alone. This had an accuracy of about 0.5, indicating a result similar to chance. This gives added evidence that sex and age don’t have much predictive power for this dataset.

However, when MMSE was added to the image array, there was a noticeable improvement. We might assume that this is the best model, but it would be unsound to use as a predictor data that has been used to define the ground truth. Hence, we cannot use a model that contains MMSE. The fact that the model improved with MMSE, however, suggests that the idea of inserting text data directly into an image array is an effective one. No literature was found discussing this technique so it remains experimental and warrants further investigation, perhaps with another dataset that has more predictive text features.

## 4.3 Why does a single slice perform better than multiple slices?

More data are usually better than less in the training of models (Chollet, 2018; Burkov, 2019). It seems counterintuitive, therefore, that taking more slices from a scan leads to reduced performance (Figure 12). However, due to the brain’s anatomy, it’s likely that slices contain different features. Figure 12 shows a definite deterioration when the 61-slice dataset is used, and looking at Figure 7 we can see that as we get towards the ends of the axial slices (i.e. towards the top and bottom of the brain) there is less/different detail in the images. It may be hypothesized that using the single middle slice is a form of feature extraction: removing features that aren’t useful in the classification task. These extraneous features may act as “noise”. There is evidence that noise in the data leads to deterioration in performance (da Costa *et al.*, 2016). This could be tested empirically by running the model on various (axial) single slices from different areas of the brain. Clinical specialists may also confirm that the “middle” part of the brain has more discriminatory features for AD.

## 4.4 Reproducibility

In this study it was noticed that running the same model on the same data gives different results.

Although one would not expect stochastic optimisation to give exact results on different runs, the Keras software documentation gives instructions on obtaining reproducible results. This involves setting various seeds and forcing the backend Tensorflow software to have a single thread and a well-defined initial state (FAQ - Keras Documentation, n.d.). These instructions were followed, but the issue persisted. The documentation mentions that a source of variability could arise from the use of a GPU and suggests using the CPU only. This was tried but proved to be unfeasible due to the extreme slowness of running CNNs on a CPU.

This issue is not discussed much in the literature although Backstrom *et al.* (2018) and Wang *et al.* (2018) take the average of several runs of their models, implying they had the same issue. Taking averages of many runs of a CNN model is also suggested by Brownlee (2017). This is the approach taken in the current study where the results given in and Table 6 are an average of 5 runs of 20 epochs each. It should be noted however that although this approach mitigates variation it doesn’t eliminate it altogether and thus comparison of various models is made difficult, especially when results are similar. However, the study is reproducible in that running the code given is expected to give *similar* results.

## 4.5 Ground-truth accuracy

MIRIAD data classifies subjects as AD or HC depending on their baseline MMSE score: a subject with a score of 26 or under is AD, while one with 27 or above is HC (Malone *et al.*, 2013). It’s interesting that subjects 246 and 251, which the CNN either misclassified or “wasn’t sure” about () are right on this borderline, and have variable MMSE scores over time sometimes putting them above, sometimes below the AD threshold (section 3.3). This may lead us to question the ground-truth (labelling) of the data since MMSE tests are just one indicator of AD and may offer only modest accuracy (Mitchell, 2009). Also, if only the first (baseline) test was used for categorisation, there are many contingent aspects that may affect the result of that one test (e.g. lack of sleep). If the ground truth is indeed questionable for borderline cases, we cannot properly assess the performance of the CNN. Perhaps this underlines the fact that diagnosis of AD needs multiple evidence sources.

## 4.6 Limitations of study and possible improvements to methodology

CNNs are a so-called “black box”: it’s impossible to explain their results. This may be unsatisfactory in a medical setting. Additionally, they require more resources than other ML algorithms, in particular, a GPU. Experiments running the models using just a CPU were abandoned due to extreme slowness; the GPU enables models to run orders of magnitude faster. Another unsatisfactory aspect of this study (and CNNs in general) may be described as parameter/hyperparameter “combinatorial explosion”. In this study we dealt with the problem by keeping most of the CNN settings constant (Table 3) and varying only the number of layers and convolutional filters per layer. There’s no easy solution to this problem with respect to CNNs. With other ML algorithms we can automate a systematic checking of hyperparameter combinations. But with CNNs this strategy can be ineffective because of the length of time a CNN model can take to run (particularly if we have several runs to mitigate reproducibility issues). Thus, in this study a pragmatic approach is taken whereby the CNN setup is based on examining past successful setups in the literature. Possibly, better results might have been obtained with more hyperparameter experimentation.

Probably the best test of the methodology (and one likely to improve the results) is to try it on larger datasets such as ADNI (Alzheimer’s Disease Neuroimaging Initiative, 2017) or OASIS (Marcus *et al.*, 2010). Another potential source of improvement is to develop a 3D CNN model, where we input the whole 3D image rather than slices. According to Payan and Montana (2015), a 3D model yielded better results than a 2D slice model, so this seems to be a fruitful exploration. Much of the literature uses pretrained CNNs (e.g. Farooq *et al.*, 2017; Chollet, 2018). These are CNNs previously trained on large datasets of images. It turns out that these CNNs are useful even if the images they are trained on are different from medical images because the features learned act as a generic model of the visual world (Chollet, 2018). Good results have been obtained from pre-trained CNNs and this is another area that could be explored.

## 4.7 Conclusion

This study obtains 89% accuracy from a CNN model predicting Alzheimer’s Disease from MRI scans. It suggests that using a single brain-slice is effective, that non-image data can be easily incorporated into a CNN and that good results can be obtained from relatively simple methodology using a relatively inexpensive standalone PC. Further work suggested to confirm and improve findings includes reproducibility research, use of other datasets, use of a 3D CNN and use of pre-trained CNNs.

# REFERENCES

Alzheimer’s Disease Neuroimaging Initiative (no date). Available at: http://adni.loni.usc.edu/ [online] (Accessed: 26 July 2019).

Alzheimer’s disease (2018). Available at: https://www.nhs.uk/conditions/alzheimers-disease/ [online] (Accessed: 24 July 2019).

Alzheimer’s Disease International (no date). Available at: https://www.alz.co.uk/ [online] (Accessed: 24 July 2019).

Anwar, S. M. *et al.* (2018) ‘Medical Image Analysis using Convolutional Neural Networks: A Review’, *Journal of Medical Systems*, 42(11), p. 226. doi: 10.1007/s10916-018-1088-1.

Ashburner, J. and Friston, K. J. (2005) ‘Unified segmentation’, *NeuroImage*, 26(3), pp. 839–851. doi: 10.1016/j.neuroimage.2005.02.018.

Backstrom, K. *et al.* (2018) ‘An efficient 3D deep convolutional network for Alzheimer’s disease diagnosis using MR images’, in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, Washington, DC: IEEE, pp. 149–153. doi: 10.1109/ISBI.2018.8363543.

Brownlee, J. (2017) ‘How to Get Reproducible Results with Keras’, *Machine Learning Mastery*, 13 June. [online] Available at: https://machinelearningmastery.com/reproducible-results-neural-networks-keras/ (Accessed: 8 August 2019).

Burkov, A. (2019) *The hundred-page machine learning book*. Self-published.

Chicco, D. (2017) ‘Ten quick tips for machine learning in computational biology’, *BioData Mining*, 10(1), p. 35. doi: 10.1186/s13040-017-0155-3.

Chollet, F. (2018) *Deep learning with Python*. Shelter Island, New York: Manning Publications Co.

da Costa, G. B. P. *et al.* (2016) ‘An empirical study on the effects of different types of noise in image classification tasks’, *arXiv:1609.02781 [cs]*. [online] Available at: http://arxiv.org/abs/1609.02781 (Accessed: 31 July 2019).

Chatterjee, S. (2017) ‘Different Kinds of Convolutional Filters‘. [online] Available at: https://www.saama.com/blog/different-kinds-convolutional-filters/ (Accessed: 24 July 2019).

FAQ - Keras Documentation (no date). [online] Available at: https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development (Accessed: 8 August 2019).

Farooq, A. *et al*. (2017) ‘A deep CNN based multi-class classification of Alzheimer's disease using MRI‘ 2017 IEEE International Conference on Imaging Systems and Techniques (IST), Beijing, pp. 1-6.  
doi: 10.1109/IST.2017.8261460

Folstein, M. F., Folstein, S. E. and McHugh, P. R. (1975) ‘“Mini-mental state”’, *Journal of Psychiatric Research*, 12(3), pp. 189–198. doi: 10.1016/0022-3956(75)90026-6.

Frisoni, G. B. *et al.* (2010) ‘The clinical use of structural MRI in Alzheimer disease’, *Nature Reviews Neurology*, 6(2), pp. 67–77. doi: 10.1038/nrneurol.2009.215.

Getsios, D. *et al.* (2012) ‘An economic evaluation of early assessment for Alzheimer’s disease in the United Kingdom’, *Alzheimer’s & Dementia*, 8(1), pp. 22–30. doi: 10.1016/j.jalz.2010.07.001.

Hosseini-Asl, E., Keynton, R. and El-Baz, A. (2016) ‘Alzheimer’s disease diagnostics by adaptation of 3D convolutional network’, in *2016 IEEE International Conference on Image Processing (ICIP)*. *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA: IEEE, pp. 126–130. doi: 10.1109/ICIP.2016.7532332.

ICD-10 Version:2016 (no date). [online] Available at: https://icd.who.int/browse10/2016/en (Accessed: 5 August 2019).

Islam, J. and Zhang, Y. (2018) ‘Brain MRI analysis for Alzheimer’s disease diagnosis using an ensemble system of deep convolutional neural networks’, *Brain Informatics*, 5(2), p. 2. doi: 10.1186/s40708-018-0080-3.

Khvostikov, A. *et al.* (2018) ‘3D CNN-based classification using sMRI and MD-DTI images for Alzheimer disease studies’, *arXiv:1801.05968 [cs]*. [online] Available at: http://arxiv.org/abs/1801.05968 (Accessed: 25 July 2019).

Lee, J.-G. *et al.* (2017) ‘Deep Learning in Medical Imaging: General Overview’, *Korean Journal of Radiology*, 18(4), p. 570. doi: 10.3348/kjr.2017.18.4.570.

Liu, J. *et al.* (2018) ‘Applications of deep learning to MRI images: A survey’, *Big Data Mining and Analytics*, 1(1), pp. 1–18. doi: 10.26599/BDMA.2018.9020001.

Liu, M. *et al.* (2018) ‘Landmark-based deep multi-instance learning for brain disease diagnosis’, *Medical Image Analysis*, 43, pp. 157–168. doi: 10.1016/j.media.2017.10.005.

Malone, I. B. *et al.* (2013) ‘MIRIAD—Public release of a multiple time point Alzheimer’s MR imaging dataset’, *NeuroImage*, 70, pp. 33–36. doi: 10.1016/j.neuroimage.2012.12.044.

Marcus, D. S. *et al.* (2010) ‘Open Access Series of Imaging Studies: Longitudinal MRI Data in Nondemented and Demented Older Adults’, *Journal of Cognitive Neuroscience*, 22(12), pp. 2677–2684. doi: 10.1162/jocn.2009.21407.

MathWorks (no date) ‘What Is a Convolutional Neural Network?‘. [online] Available at: https://uk.mathworks.com/solutions/deep-learning/convolutional-neural-network.html (Accessed: 24 July 2019).

Mitchell, A. J. (2009) ‘A meta-analysis of the accuracy of the mini-mental state examination in the detection of dementia and mild cognitive impairment’, *Journal of Psychiatric Research*, 43(4), pp. 411–431. doi: 10.1016/j.jpsychires.2008.04.014.

Neuroimaging in Python - Pipelines and Interfaces — nipy pipeline and interfaces package (no date). [online] Available at: https://nipype.readthedocs.io/en/latest/ (Accessed: 27 July 2019).

Payan, A. and Montana, G. (2015) ‘Predicting Alzheimer’s disease: a neuroimaging study with 3D convolutional neural networks’, *arXiv:1502.02506 [cs, stat]*. [online] Available at: http://arxiv.org/abs/1502.02506 (Accessed: 24 July 2019).

Research Imaging Institute — Mango (no date). [online] Available at: http://ric.uthscsa.edu/mango/ (Accessed: 27 July 2019).

Sharma, A. (2017) ‘Convolutional Neural Networks in Python with Keras', *DataCamp Community*. [online] Available at: https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python (Accessed: 8 August 2019).

Song, S., Zheng, Y. and He, Y. (2017) ‘A review of Methods for Bias Correction in Medical Images’, *Biomedical Engineering Review*, 3(1). doi: 10.18103/bme.v3i1.1550.

SPM12 - Statistical Parametric Mapping (no date). [online] Available at: https://www.fil.ion.ucl.ac.uk/spm/software/spm12/ (Accessed: 27 July 2019).

UCL (2018) *Minimal Interval Resonance Imaging in Alzheimer’s Disease (MIRIAD)*, *Dementia Research Centre*. [online] Available at: https://www.ucl.ac.uk/drc/research/methods/minimal-interval-resonance-imaging-alzheimers-disease-miriad (Accessed: 27 July 2019).

Waldemar, G. *et al.* (2007) ‘Recommendations for the diagnosis and management of Alzheimer’s disease and other disorders associated with dementia: EFNS guideline’, *European Journal of Neurology*, 14(1), pp. e1–e26. doi: 10.1111/j.1468-1331.2006.01605.x.

Wang, S.-H. *et al.* (2018) ‘Classification of Alzheimer’s Disease Based on Eight-Layer Convolutional Neural Network with Leaky Rectified Linear Unit and Max Pooling’, *Journal of Medical Systems*, 42(5), p. 85. doi: 10.1007/s10916-018-0932-7.

Xu, B. *et al.* (2015) ‘Empirical Evaluation of Rectified Activations in Convolutional Network’, *arXiv:1505.00853 [cs, stat]*. [online] Available at: http://arxiv.org/abs/1505.00853 (Accessed: 27 July 2019).

# ACKNOWLEDGEMENTS

I would like to thank my dissertation supervisor, Dr Holger Kunz, for suggesting the idea of insertion of non-image data directly into the image tensor.

Data used in the preparation of this article were obtained from the [MIRIAD database](http://www.ucl.ac.uk/silva/drc/research/miriad-scan-database). The MIRIAD investigators did not participate in analysis or writing of this report. The MIRIAD dataset is made available through the support of the [UK Alzheimer's Society](http://www.alzheimers.org.uk/) (Grant RF116). The original data collection was funded through an unrestricted educational grant from GlaxoSmithKline (Grant 6GKC).

# Appendix 1: Python code

# Appendix 2: Hardware and software used

**Hardware**

Standalone PC from Scan computers ([www.scan.co.uk](http://www.scan.co.uk/)) with the following specification:

Architecture: x86\_64

Chip/memory: Intel Core i9 with 64 Gb of memory

Hard disk: Samsung 1 Terabyte SSD

GPU: NVIDIA GE Force RTX 2080 TI

Operating System: Ubuntu 18.04.02 (Bionic)

**Software**

Statistical Parametric software v. 12 (SPM 12)

Matlab v. 2019a

Python v. 3.7.3

Python libraries:

* Numpy v. 1.16.2
* Pandas v 0.24.2
* Keras v. 2.2.4
* Tensorflow v. 1.13.1
* Nipype v. 1.2.0
* Nibabel v. 2.4.1

# Appendix 3: CNN map of best model found (64, 64, 64)

