Dementia Risk Detection using Deep Neural Networks

Danai-Georgia Sakelliadou, Shivani Hegde
Department of Computer Science, Boston University, USA
dgs2012@bu.edu, svhegde@bu.edu

CAS CS 523 Deep Learning
Professor Sang Chin
Boston University

Abstract

Two baseline Convolutional Neural Network (CNN) architectures were used in the development of this project with the goal of predicting the risk of an individual who has mild cognitive impairment developing Alzheimer's disease with higher precision. One of the CNNs is a custom structure that was created by us and the other CNN model used is AlexNet, originally created by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoggrey Hinton [9]. For the input dataset, we used Magnetic Resonance Imaging (MRI) .jpg images from Kaggle containing 5121 images in total with imbalanced distribution of 4 classes. Due to the imbalanced dataset, we used a Receiver Operations Characteristic (ROC) Area Under Curve (AUC) metric. After putting the Kaggle dataset [7] through both models, the custom CNN model resulted in a test loss of 0.8208 and and AUC metric of 0.8742 compared to the AlexNet model test loss of 1.0465 and AUC of 0.7796.

1 Introduction

1.1 Alzheimer's Disease

Alzheimer's Disease is a chronic neurodegenerative disease and it's considered to be the most common cause of dementia globally [2]. It affects the quality and the safety of the patient due to memory degradation and cognitive function impairment and it's the 5th leading cause of death for people over 65 years old [16]. It affects over 5.8 million U.S adults each year and with an annual total treatment cost estimating at 305 billion USD [16], this disease's impact comprises a major concern both for quality of life as well as socioeconomic development. Currently, there is no way to prevent or cure Alzheimer's disease but the earlier an individual can receive the proper diagnosis, the sooner that treatment can be administered. However, to detect AD, there is no standardized testing [4]. Magnetic Resonance Imaging (MRI) images of the brain are used to classify healthy brains from those that have progressive mild cognitive impairment based on the structural atrophy of the brain because of Alzheimer's disease [3].

1.2 Application of Deep Learning

This is where our project comes in by using deep neural network structures to predict the risk of developing Alzheimer's disease with symptoms of mild cognitive impairment. In the case of our project, because the MRI images capture the brain activities that display varying cranial functions, image classification can be used to identify and categorize these activities to different levels of cognitive impairment - including none. Building classifiers using deep learning helps idntify the important structural differences in the regions such as the entorinhal cortex and the hippocampus

entorhinal cortex between the brain with AD and healthy brain [2]. Particularly, Convolutional Neural Networks (CNNs) have performed significantly better in medical image analysis. Medical professionals can take aid of classification deep neural networks to make decisions for their subjects or patients on a level of more precision. Our first step towards that direction has been building a CNN structure with the hopes of using it as a baseline to precisely classify MRI images.

2 Background

‘

2.1 Magnetic Resonance Imaging Images

High resolution MRI images which have been proved to yield the best predictions when combined with deep learning applications. This is because through brain MRIs, most of the intermediate states of Alzheimer's disease (c-MCI,s-MCI) along with their corresponding structural differences can be visualised clearly and thus provide us with a trustworthy predictive model. MRI can provide valuable information about tissue structures including localization, size and shape so a deep learning guided process that takes account into these details can further increase the precision of the diagnosis [10].

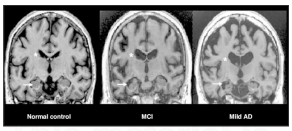

Figure 1. Images of brain MRI scans.

2.2 CNNs as the main Deep Learning architecture for AD risk prediction

Recent research breakthroughs have showcased the importance of using CNN to create a model for predicting Alzheimer's Disease. Convolution Neural Network (CNN) structures are designed to recognize images by having convolutions inside, that see the edges of recognized objects on the image [2]. The standard architecture of a CNN comprises of the input layer,followed by alternating convolutional and max pooling layers which are then followed by dense (fully connected) layers and finally the output layer. It's typical to add use a Rectified Linear Unit (ReLU) with this structure, which is the activation function we have been using in this project. The basic advantage of CNNs in general, is their ability of faster generalization as they decrease the number of learnable parameters through their properties [5].More specifically, in AD research, besides the plus of dealing well with unseen patient data, they also give us the ability of faster classification by applying Transfer Learning, i.e using pre-trained convolutional networks, to accelerate our computations and thus, our prediction. In our model, the output of both of the CNNs we created provide us with a classification result between 4 possible states of Dementia(Non-Demented, Very Mild Demented, Mild Demented, Moderate Demented). – ref

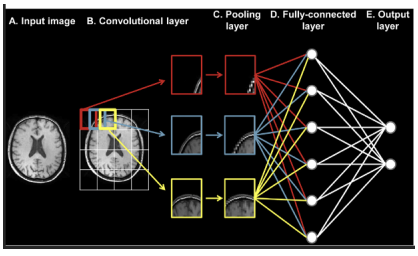

Figure 2. Example of a Convolutional Neural Network structure.

2.2.1 Transfer Learning

One of the goals of this project was to compare the performance of different pre-trained CNN structures with the same MRI images dataset from Kaggle and compare the results to the ones from the custom CNN. Fine-tuning a pre-trained model using particular samples from the relevant dataset

is useful when the size of the available dataset is relatively small compared to what is required to train a custom model to great precision. However, this may come to be troublesome because due to the nature of the pre-trained model's dataset of generic images. This may affect the performance of the classification model due to the lack of similarity between the custom dataset and the pre-trained model's dataset.

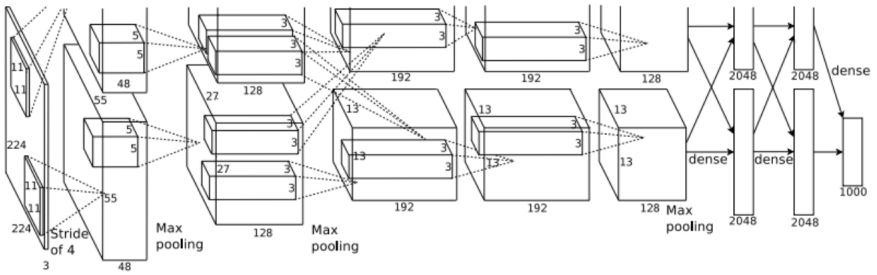

Figure 3. Illustration of AlexNet's architecture. Image credits to Krizhevsky et al., from the original paper of AlexNet.

3 Methodology

3.1 Kaggle Dataset

The Kaggle dataset [7] used as input for the CNNs consisted of a total of 5121 MRI .jpg images with an 80/20 split between the training and validation set and a separate testing dataset consisting of 80 images. In both the training/validation and testing set, there are 4 classes as previously mentioned; Very Mild Demented, Mild Demented, Moderate Demented, and Non Demented. All the images are randomly transformed and shuffled and used as input into the neural network.

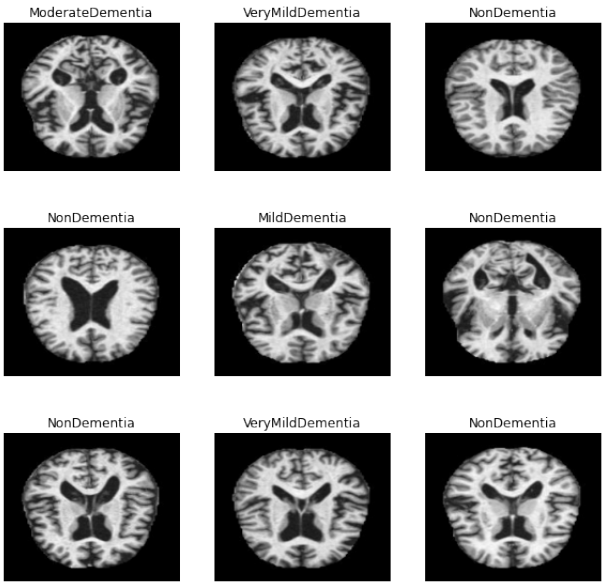

Figure 4. Images from the Kaggle MRI dataset with their respective labels.

3.2 Custom CNN

The custom CNN architecture was inspired from two sources that both used their models to be able to classify Alzheimer's disease MRI images.

The first source was used as the main structure of the code and it was from a Kaggle tutorial guided by Amy Jang [8]. It loaded the dataset using TensorFlow's

`keras.preprocessing.image_dataset_from_directory()` module. The training dataset was streamlined into using one-hot encodings in order for the model to categorically interpret the data rather than continuously. The model is built using Tensorflow's Keras to call the layers of a CNN easily. This Kaggle tutorial used blocks of convolutional and dense layers that had took in varying dimensional filters and the dense blocks additionally took dropout rate as input. The convolutional block consists of two `SeparableConv2D()` layers with `ReLU` activation then normalized through `BatchNormalization()` and ends with a `MaxPool2D()` layer. The dense block has one dense layer activated by `RelU`, normalized with `BatchNormalization()` and ends with a `Dropout()` layer that takes as input the dropout rate provided by the user.

The second source that inspired the customized CNN model in this project derived from an Alzheimer's disease classification paper using Fully Connected Networks (FCNs) alongside Multilayer Perceptrons (MLPs) [12]. The framework of this paper connects an FCN to an MLP in order to perform binary classification and predict the status of Alzheimer's disease in a subject through probability maps. The 3D CNN model consisted of many repeated blocks of [Conv3D layer, MaxPool3D, BatchNormalization, Leaky ReLU, Dropout] before flattening and then patterning Dropout layers, Dense layers, and Leaky ReLU.

Putting the two structures together, is the customized CNN model built in this project. the customized CNN structure consists of 3 blocks of `[Conv2D(), MaxPool2D(), BatchNormalization(), LeakyReLU(), Dropout()]` where the 2D convolutional layers are activated by ReLU - all taking in 16 channels as input, the batch normalization layers have momentum ranging from 0.6 to 0.7 depending on the runs and epsilon always at 10^{-5}, and the dropout layers are dropping at a rate of 20%. After these blocks, then there are 3 convolutional blocks with 32, 64, and 128 square dimensional filters respectively. The dropout rate hits at 20% , followed by a convolutional block with 356 filters as input. The layers get flattened and 3 dense blocks applied subsequently; with 512 unit dimensional outputs and 70% dropout rate, 128 outputs and 50% dropout rate, and 64 dimensions and 30% dropout rate respectively.

The metrics chosen for this model is Area Under Curve (AUC) Receiver Operating Characteristics (ROC) curve. The Kaggle dataset is imbalanced given that for the 4 classes of images, the distribution is: `[717, 52, 2560, 1792]`, so we cannot choose the accuracy, the most conventional metric. Briefly, ROC AUC essentially provides a score where if the score is closer to 1, that means the chances of the model distinguishing different classes is higher. The closer it is to 0, the greater the indication that the model is not able to distinguish between different classes. At a score of 0.5, this would indicate that the model is classifying images rather randomly.

To train the model for the dataset, callbacks from TensorFlow help in different ways such as to adjust the learning or to stop the model once it no longer makes significant steps in the loss reduction, or even to set checkpoints where the model is saving the best version of itself in a .h5 file. An H5 file stores data in a hierarchy; in this scenario it saves the weights of the model that in combination are making the best predictions. All 3 of these callbacks are used in the code for both the custom CNN as well as the AlexNet model.

3.3 AlexNet

In order to make comparisons between the architectures, the only change in the code was made to the model function that contained all the layers of the structure - everything else was left intact. AlexNet consists of 4 blocks of `Conv2D(), BatchNormalization(), MaxPool2D`, 2 blocks of `[Conv2D(), BatchNormalization()]`, flattened layer, 2 blocks of `Dense(), Dropout()` and ending with a Dense layer with softmax activation**REFERENCE - ALEXNET**. The convolution layers varied in many different dimensional outputs, and the kernel sizes got smaller through the layers.

4 Results

4.1 Custom CNN

As we trained the model using the Kaggle dataset, we experimented with the momentum inside the batch normalization layer, varying it from 0.6 to 0.7 and adjusting the initial learning rate to be from 0.01 and going down to 0.003 as the momentum increased. The effect of adding and removing

several layers of convolutional blocks was much less effective compared to the difference it made by simply tweaking momentum and the initial learning by fractions. The training estimated time does not change with these minor adjustments either so even if there is a model that works efficiently in a timely manner, it does not matter if it does not perform nearly as well when it comes to minimizing loss and maximizing ROC AUC values.

Optimizing the training dataset does not necessarily mean that the test loss will perform proportionately and that is evident in the following graphs and epochs in the iteration comparisons. Comparing the two runs of the same custom CNN model, despite the fact that the training epochs in Figure 8 for the second run performed >35% better than in Figure 6, the test loss shown in Figure 7 for the first run is still 6% better than the test loss shown in Figure 10. The AUC metric for the first run is also better at The spikes in the validation error point towards there being some overfitting occurring however, with the validation error being lower in the second run in Figure 8, this is some unknown type of fitting where the model is better at predicting the unknown than what it has learned. This provokes the thought that perhaps the data splitting method may need to be tweaked or randomized further, or the model requires more data to be streamlined in to make better calls.

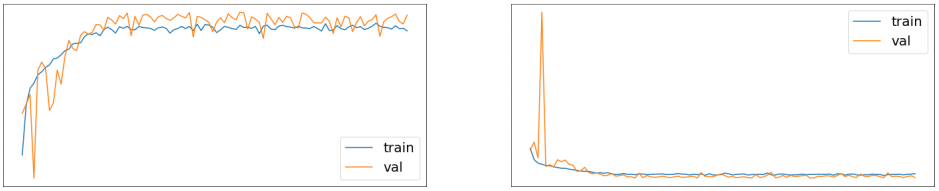

Figure 5. Although the figures do not show it here for the run with the test loss of 0.8208, the left graph shows the model's ROC AUC metric with the y-axis being the loss values for the training and validation sets. The right graph shows the model's loss graph with the y-axis being the loss values after each iteration.

```
Epoch 98/100
257/257 [==============================] - 19s 73ms/step - loss: 0.7567 - auc: 0.8938 - val_loss: 0.7321 - val_auc: 0.8995
Epoch 99/100
257/257 [==============================] - 19s 74ms/step - loss: 0.7671 - auc: 0.8932 - val_loss: 0.7421 - val_auc: 0.8970
Epoch 100/100
257/257 [==============================] - 19s 74ms/step - loss: 0.7719 - auc: 0.8905 - val_loss: 0.7190 - val_auc: 0.9059
```

Figure 6. Custom CNN training run for test loss = 0.8208 and AUC = 0.8742.

```
80/80 [==============================] - 1s 16ms/step - loss: 0.8208 - auc: 0.8742
```

Figure 7. Custom CNN test run for test loss = 0.8208 and AUC = 0.8742.

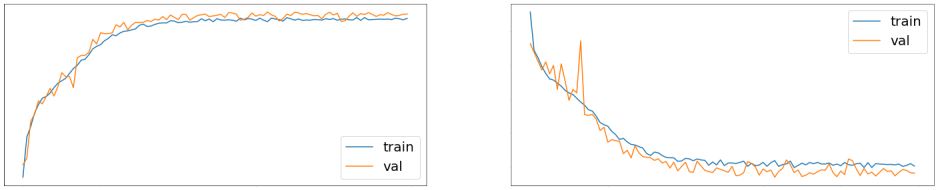

Figure 8. Also here, for the run with the test loss of 0.8811, the left graph shows the model's ROC AUC metric with the y-axis being the AUC values for the training and validation sets. The right graph shows the model's loss graph with the y-axis being the loss values after each iteration.

```
Epoch 98/100
257/257 [==============================] - 19s 74ms/step - loss: 0.4060 - auc: 0.9695 - val_loss: 0.3790 - val_auc: 0.9746
Epoch 99/100
257/257 [==============================] - 19s 74ms/step - loss: 0.4279 - auc: 0.9655 - val_loss: 0.3656 - val_auc: 0.9764
Epoch 100/100
257/257 [==============================] - 19s 75ms/step - loss: 0.4161 - auc: 0.9678 - val_loss: 0.3640 - val_auc: 0.9768
```

Figure 9. Custom CNN training run for test loss = 0.8811 and AUC - 0.8860.

```
80/80 [==============================] - 1s 16ms/step - loss: 0.8811 - auc: 0.8860
```

Figure 10. Custom CNN test run for test loss = 0.8811 and AUC = 0.8860.

4.2 AlexNet

In order to take advantage of the usage of pre-trained CNNs, AlexNet was used in this project in the form of Transfer Learning. The same loss function, Categorical Cross Entropy was used in both CNN models so this stayed the same for AlexNet. In the run that we performed for AlexNet, the test loss was higher by >20% at 1.0365 and the AUC metric was 0.7796 for the AlexNet run, which means that the AlexNet model currently does not distinguish between the 4 different classes as well as the custom CNN model. Due to the early stopping callback from TensorFlow being activated on the AlexNet run, the epochs had stopped early and had run through the test dataset so it is currently not known as to what the final test loss after 100 epochs would be, which would have been a more fair comparison between the two models.

Unfortunately, for the time being, AlexNet training/validation graph could not be acquired due to technical issues with the Jupyter notebook that ran the model.

```
Epoch 15/100
257/257 [==============================] - 212s 825ms/step - loss: 1.0357 - auc: 0.7832 - val_loss: 1.0406 - val_auc: 0.7764
Epoch 16/100
257/257 [==============================] - 211s 822ms/step - loss: 1.0379 - auc: 0.7785 - val_loss: 1.0404 - val_auc: 0.7764
Epoch 17/100
257/257 [==============================] - 211s 823ms/step - loss: 1.0384 - auc: 0.7802 - val_loss: 1.0406 - val_auc: 0.7765
```

Figure 11. AlexNet training run for test loss = 1.0365 and AUC = 0.7796.

```
321/321 [==============================] - 45s 139ms/step - loss: 1.0365 - auc: 0.7796
```

Figure 12. AlexNet test run for test loss = 1.0365 and AUC = 0.7796.

5 Conclusion and Outlook

5.1 Summary

The main reason for us to begin with creating a baseline CNN model for Dementia risk prediction is that it will give us the potential to work on several ideas in order to improve the prediction algorithm. After putting the Kaggle dataset through both models, the custom CNN model resulted in a test loss of 0.8208 and and AUC metric of 0.8742 compared to the AlexNet model test loss of 1.0465 and AUC of 0.7796. Our major concern and our priority is the increase the variance of our model's input data type and to try the ConvLSTM structure [13].

5.2 Future Implementations

This means that our future plans have to do with receiving input from the ADNI dataset (.nii) instead of Kaggle images (.jpg). This will take us a step further to using more of ADNI's data like biomarkers, genetic variants and EHR, combining it MRIs and feeding that to our model to train it more efficiently and make it more foolproof [12].

In continuation to that, we envision of creating a deep learning framework strong enough to bridge the gap between preclinical stages of dementia, that cannot be visualized, and the actual development of the disease's symptoms and its subsequent cognitive decline.

For that purpose, we thought of combining biomarkers from different activities of life, that may seem to be of minor importance but combined with our pre-trained and already structured model, may actually give us the prediction we need, much earlier during the patient's life, than our current predictive systems can. An example of these is: using GPS driving as a digital biomarker [6], trained NLP model outputs out of conversational transcripts as ancillary input [11], or even creating a differential diagnosis algorithm between Alzheimer's and Parkinson's from accelerometers [14].

6 Acknowledgements

We would like to thank Amy Jang from the Kaggle tutorial [8] for the framework of the code that we used.

References

[1] Al-Khuzhaie, F. E. K., et al.: Diagnosis of Alzheimer Disease Using 2D MRI SLices by Convolutional Neural Network. Applied Bionics and Biomechanics, Hindawi (2021) doi.org/10.1155/2021/6690539

[2] Alzheimer's Disease and Healthy Aging Data. Centers for Disease Control and Prevention. https://chronicdata.cdc.gov/Healthy-Aging/Alzheimer-s-Disease-and-Healthy-Aging-Data/hfr9-rurv

[3] Amoroso, N., La Rocca, M., Bruno, S., Maggipinto, T., Monaco, A., Bellotti, R., et al.: Brain structural connectivity atrophy in Alzheimer's disease, preprint arXiv:1709.02369 (2017)

[4] Barrett, E., Burns, A.: Dementia Revealed. What Primary Care Needs to Know. Department of Health UK (2014)

[5] Basaia, S. et al.: Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks. NeuroImage Clinical, Elsevier (2018) https://www.sciencedirect.com/science/article/pii/S2213158218303930

[6] Bayat, S., Babulal, G.M., Schindler, S.E. et al.: GPS driving: a digital biomarker for preclinical Alzheimer disease. Alz Res Therapy 13, 115 (2021) https://doi.org/10.1186/s13195-021-00852-1

[7] Dubay, S.: Alzheimer's Dataset (4 classes of Images), Version 1 (2019). https://www.kaggle.com/tourist55/alzheimers-dataset-4-class-of-images

[8] Jang, Amy.: Alzheimer MRI Model + Tensorflow 2.3 Data Loading. Kaggle (2020) https://www.kaggle.com/amyjang/alzheimer-mri-model-tensorflow-2-3-data-loading/data

[9] Krizhevsky, A. et al.: ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems, Vol 1, 1097-1105 (2012)

[10] Liu, J. et al.: Complex brain network analysis and its applications to brain disorders: a survey. Complexity, vol. 2017, Article ID 8362741, 27 pages (2017)

[11] Mahajan AD Paper: Mahajan, P., Baths, V.: Acoustic and Language Based Deep Learning Approaches for Alzheimer's Dementia Detection From Spontaneous Speech. Front. Aging Neurosci. 13:623706 doi: 10.3389/fnagi.2021.623607

[12] Qiu S. et al.: Development and validation of an interpretable deep learning framework for Alzheimer's disease classification. Oxford Academic: Brain, Vol 143, Issue 6 (2020) https://academic.oup.com/brain/article/143/6/1920/5827821

[13] Rahman, F. et al.: Prognostic Prediction and Classification of Alzheimer's Neuroimage Sequences with a Convolutional LSTM Network. Yale University (2020) https://github.com/Yale-Deep-Learning-for-Alzheimers/Alzheimers-DL-Network

[14] Schwab, P. et al.: Automated Extraction of Digital Biomarkers for Parkinson's Disease. 10th Annual RECOMB/ISCB Conference on Regulatory Systems Genomics with DREAM Challenges (2017) doi.org/10.3929/ethz-b-000224487

[15] Wang, Y. R.: Convolutional Neural Networks for Alzheimer's Disease Prediction using Brain MRI Image. (2018) https://github.com/wangyirui/AD_Prediction

[16] Wong, W.: Economic Burden of Alzheimer Disease and Managed Care Considerations. The American Journal of Managed Care, US National Library of Medicine (2020) https://pubmed.ncbi.nlm.nih.gov/32840331/