CS230

Interpretable 2D and 3D Convolutional Neural Networks for Alzheimer's Disease in Brain Scans

Michael Wang
Department of Computer Science
Stanford University
mzwang14@stanford.edu

Abstract

Alzheimer's disease is an incurable brain disorder as well as the 6^{th} leading cause of death in the USA[1] . Magnetic Resonance Imaging(MRI) is a technique used to form pictures of anatomy within the body, and can be used for brain scans. In this project, I implemented multiple 2D and 3D models in order to try and detect Alzheimer's from MRI scans, and implemented a paper called Grad-CAM[2] in order to determine what led to the diagnosis.

1 Introduction

Detecting Alzheimer's in a timely manner is very important in order for patients to receive treatment and slow down the progress of the disease. However, with purely behavioral metrics, it can be easy to conflate Alzheimer's with standard memory loss, and using MRI scans to diagnose the disease is still an active research problem, as early Alzheimer's can seem normal on an MRI[3]. This has been my motivation to attack this problem using machine learning techniques, and to implement visualization techniques in order to determine the changes in the brain that result in Alzheimer's.

I trained both 2D and 3D convolutional neural networks using Keras[4] and the OASIS[5] dataset, where the input to the 2D models is the horizontal middle slice of the MRI scan, and the input to the 3D models is the entire scan. For each input, the model outputs a diagnosis for the patient as a binary classification problem. I then used Grad-CAM to output a class activation map for each diagnosis. I trained multiple different architectures using an end-to-end approach, and achieved slightly worse performance than previous research efforts in the area. The best 2D model was a fine-tuned Inception V3[6] network with ImageNet[7] weights, achieving 76% accuracy on the testing set and a .78 F1 score.

2 Related work

2.1 2D Detection using MRI Scans

The following paper[8] provides an overview of various deep learning techniques used for Medical MRI imaging, including different network architectures, as well as medical techniques such as patches and segmentation.

2D detection using the OASIS dataset was done to classify various stages of Alzheimer's disease[9], where they were able to achieve 91% accuracy classifying various states of the disease by implementing their own architecture and using 2D patches, which is less of an end to end approach as mine. Another study[10] used a sparse autoencoder along with cross domain features to represent MRI data, trained on the ADNI[11] dataset, achieving 95% accuracy.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2.2 3D models and Interpretability

The following study[12] implements various 3 dimensional models and implements a 3D Grad-CAM for visual explanations, with the two best performing models being 3d-ResNet[13] and 3D-VGGNet[14]. They also used the ADNI dataset.

3 Dataset and Features

3.1 Dataset Description

For this project, I used the OASIS(Open Access Series of Imaging Studies) dataset, which provides a dataset of MRI scans of the brain. There are 1098 patients, with 2118 MRI sessions, with a total of 3254 scans that I used, as each session often had more than one scan. There were a total of 609 cognitively normal patients, and 489 patients with varying states of cognitive decline. For the purposes of this project, I treated all patients who were evaluated as cognitively impaired during a psychiatric evaluation as demented. In total, I ended up with 597 scans labelled as demented, and 2657 scans labelled as normal cognition. To remedy the class imbalance during my training, I set the class weight of demented to .2, and normal cognition to .8 during training.

	max CDR					
min CDR	0	0.5	1	2	3	Grand Total
0	609*	192	39	12	2	854
0.5		66	61	45	5	177
>1			31	31	5	67
Grand Total	609	258	131	88	12	1098

**Unchanged CDR = 0 represents cognitively healthy population*

Figure 1: Table with the distribution of patients and their states of cognitive decline.

3.2 Data Collection

I started by crawling the dataset and downloading the raw scans and CSV files containing the labels,which are in Nifti file formats. Each file contains 256 2D slices of the brain in order to represent a 3D scan. The scans contain the patient ID and the date the scan was taken. The CSV files contain a psychiatric evaluation as well as the date of the evaluation. In order to label the scans, I wrote a script to parse the scans and label them with the result of the closest evaluation. I stored the results in a JSON file. I used roughly a 80-10-10 train/validate/test split. One issue that often happens chose to use the scans as independent samples, which results in data leakage - many of the same subjects in the training set appear in the testing set.

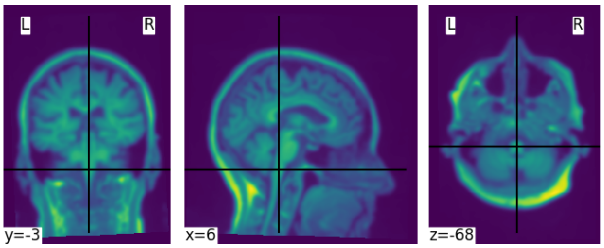


Figure 2: A brain scan from three different angles

In order to remedy this, I wrote scripts to parse the patient data, and labelled each patient with whether they end up contracting Dementia, and organized the scans by ensuring all the scans for a single patient are in the same set. Because each patient has a varying number of scans, this did not result in exactly a 80-10-10 training/validation/test split.

3.3 Preprocessing

I used a tool called N4BiasFieldCorrection[15] for normalization, which is a popular method for correcting low frequency non-uniformity present in MRI image data. It is a variant of the popular nonparametric nonuniform intensity normalization (N3) algorithm. I also resized the images to make sure they were all the same size, and stripped the skulls from the image. Afterwards, I used the custom Keras preprocessing function for each model.

For my 2D classifiers, I took the middle horizontal slice for each scan as a grayscale .jpg file. For the 3D classifier, I organized my normalized scans directly into directories and used them as inputs to my classifiers. I used Keras's ImageDataGenerator for data augmentation - the only augmentation technique used was shear range and zoom, as I felt like the other techniques such as image flips would not be relevant to brain scans.

4 Methods

4.1 2D Baseline

I began with a baseline Convolutional Neural Network that used a 32 filter convolution layer(5x5) followed by a 64 filter convolution layer(5x5) with max pooling(2x2) layers between each. This was connected to a fully connected layer(500) followed by a sigmoid activation. I used binary cross entropy loss and an Adam optimizer with the default parameters.

4.2 2D Transfer Learning

I fine-tuned several pre-trained model architectures. I tried the VGG16, VGG19, DenseNet121[16], DenseNet169, DenseNet201, ResNet50, Inception V3[17], and MobileNet V2[18], with imagenet weights. For each model, I used the custom preprocessing function for that model. I replaced the fully connected layers with a Flattening layer, and four dense layers with 1024, 1024, 512, and 1 nodes respectively, with Batch Normalization and a Dropout layer between. I used a sigmoid activation at the end as I am doing binary classification, and the binary cross entropy loss function as well as an Adam optimizer. I started out by training them with a dropout rate of .4 for each layer, and the default Adam parameters to find the best performing architectures.

After I found the best performing architectures, I retrained them using different hyperparameters and data augmentation, which will be discussed in the results section.

4.3 3D models

For the 3D CNN base model, I implemented the VoxCNN from the ground up. The architecture is shown below. In the original implementation, there is a batch normalization layer and a dropout of .7 between each of the dense layers except for the last one.

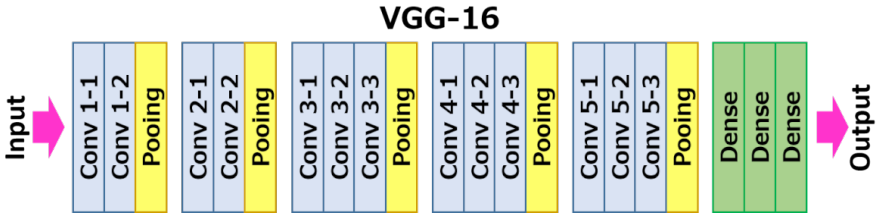


Figure 3: 3D VGG16 architecture

It was simply infeasible to load all of the 3D data into memory and then run the network on it so I took several steps in order to remedy this. First, I read all of the Nifti files using a tool called nibabel, loaded them into Numpy arrays, and saved them as .npy files. I wrote a generator to read these and feed them into the model. I then had a problem with the batch size. When the batch size was larger than 5, I wasn't able to run the model due to out of memory errors, so I wasn't able to ensure that each batch contained data points from each class. Using a batch size of 4 and 5 epochs, this model took about 4 hours to train.

4.4 Grad-CAM

Grad-CAM uses the gradient information going into the final convolutional layers to determine a class activation mapping. It weighs every channel by the gradient of the final class with respect to that channel. This "activates" different channels based on how much those channels affect the output class.

$$S^c = \frac{1}{K} \sum_i \sum_j \sum_k \frac{\partial y^c}{\partial A^j_{ik}} A^j_{ik} \tag{1}$$

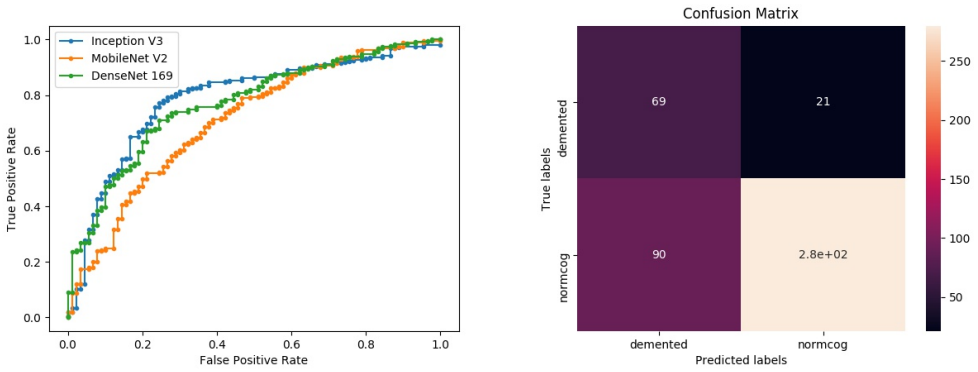
Here, S^c is the global average pooling over two dimensions i and j for the gradient of the output class y^c with respect to the feature map A^j_{ik} - we multiply this by the channel to get the class activations. Unlike normal class activation maps, this does not require retraining the network as the class activation is determined before the flattening layer. By using Grad-CAM along with training multiple models, you can ensure that classifiers are predicting classes for the correct reasons without needing to retrain the network. I think this is especially important for detecting Alzheimer's - there could be many mislabels early with behavioral evaluations. Ensuring classifiers are identifying Alzheimer's through the correct brain regions could help mitigate this.

5 Experiments/Results/Discussion

5.1 2D Models

The baseline Convolutional model did not perform well. After training for 25 epochs, it was not able to determine any distinction between the classes - I was not able to get it to predict anything other than normal cognition every time.

The three models that performed the best were Inception V3, MobileNet V2, and DenseNet 169. I believe this may be due to the smaller size of the networks compared to the VGG models which did not perform well, combined with having less data. I tuned the following hyperparameters: dropout rate of the fully connected layers, number of frozen layers, and learning rate using a train-dev set split with Keras's ImageDataGenerator, and took the best performing model on the validation data. I chose to tune the dropout rate to find the mix between overfitting and being too simplistic - my models at first had the tendency to either overfit to the training data, or to predict the same class for everything. I consistently found that freezing layers resulted in worse performance - I believe this is because brain scan features might be different from ImageNet at a low level, and modifying all the weights is helpful. Additionally, my jpg files were greyscale, while ImageNet is RGB.

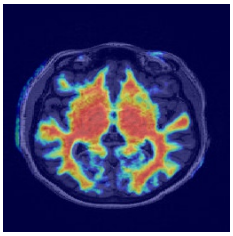


(a) ROC Curves (b) Confusion Matrix

Figure 4: ROC Curves and Confusion Matrix for InceptionV3

The results are above. The AUC for Inception V3, MobileNet V2, and DenseNet 169 are .79, .71, and .77 respectively. The best network(Inception V3) was able to correctly classify 76% of normal cognition scans, and 77% of demented scans, with a F1 score of .78. To the right is a class activation map using Grad-CAM of an Alzheimer's patient. I was not able to use it to identify whether it was predicting the right areas due to lack of domain knowledge. This would be good future work.

Figure 5: Grad-CAM output



Below is a table with the best hyperparameters and results for the three networks. I trained the models using both data augmentation and without. While the data augmentation resulted in better results for DenseNet and MobileNet, the differences were very very small, and on average models performed worse. I believe this is because Alzheimer's creates such small variations in the brain, that modifying the scans can result in scans that don't accurately mimic the correct label, resulting in a situation similar to systematic mislabeling.

	DenseNet 169	Inception V3	MobileNet V2
Best Dropout Rate	.5	.4	.6
Best Learning Rate	.001	.001	.0001
Data Augmentation	Yes	No	Yes
Test Accuracy Avg(weighted)	.74	.76	.75
Test Precision Avg(weighted)	.83	.83	.83
Test Recall Avg(weighted)	.74	.76	.75
Test F1 Avg(weighted)	.77	.78	.77
Test AUC	.81	.8	.76

5.2 3D Models

The VoxCNN trained from scratch did not perform well - I was not able to get it to perform better than random.

First, the inability for each batch to contain data points from both classes would interfere with gradient descent's optimization - I found that at the end, the classifier predicted Normal Cognition for everything with very high confidence, likely due to the distribution of data in each batch. Additionally, I was only able to train the network for 5 epochs, which is rather small for such a large number of parameters. Finally, due to the amount of time it took to train the model, I was not able to perform any search over the hyperparameter space.

6 Conclusion/Future Work

I tried several different deep learning algorithms in order tackle the problem of Alzheimer's detection in MRI scans and implemented Grad-CAM to visualize the results. The Inception net V3 performed the best, with 77% accuracy, an F1 score of .7 and .79 AUC. The 3D VoxCNN did not perform well. With more computational resources and more data, I believe that the 3D network could be improved by allowing a greater batch size, more hyperparameter tuning, and longer training. Additionally, I did an end to end approach. I believe that taking some intermediary steps could have resulted in some better results. Finally, I think additional work in utilizing Grad-CAM output in order to determine the parts of the brain that are indicative of a diagnosis would be useful.

References

[1] Alzheimer's facts and figures. https://alz.org/alzheimers-dementia/facts-figures.

[2] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[3] Radiological Society of North America, Rsna, and American College of Radiology. Alzheimer's disease.

[4] Keras: The python deep learning library.

[5] Oasis brains. Oasis: Longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer's disease. http://www.oasis-brains.org.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[7] Imagenet. http://www.image-net.org.

[8] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri, Dec 2018.

[9] Jyoti Islam and Yanqing Zhang. Brain mri analysis for alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks, May 2018.

[10] Ashish Gupta, Murat Seçkin Ayhan, Anthony S. Maida, Ashish Gupta University of Lousiana, University of Lousiana, Murat Seçkin Ayhan University of Lousiana, and Anthony S. Maida University of Lousiana. Natural image bases to represent neuroimaging data, Jun 2013.

[11] Alzheimer's disease neuroimaging initiative. http://adni.loni.usc.edu/.

[12] Chengliang Yang, Anand Rangarajan, and Sanjay Ranka. Visual explanations from deep 3d convolutional neural networks for alzheimer's disease classification. *CoRR*, abs/1803.02544, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[15] Nicholas J Tustison, Brian B Avants, Philip A Cook, Yuanjie Zheng, Alexander Egan, Paul A Yushkevich, and James C Gee. N4itk: improved n3 bias correction, Jun 2010.

[16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.