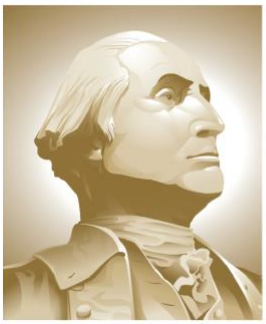


THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Capstone Report - Spring 2022

# Augmenting and Ensembling CNN Models to Increase the Accuracy of Alzheimer's Disease MRI Classification

Kristin Levine

Advisor: Amir Jafari
The George Washington University, Washington DC
Data Science Program

Abstract

Lack of data is an on-going problem in Alzheimer's disease (AD) research. AD can be hard to diagnose and even experienced neurologists struggle to read MRIs accurately. While many studies have used CNN models to classify MRI images, this study compares flipping and shifting images to determine which type of data augmentation is most helpful. This study also compares three different pre-trained CNN models (ResNet50, VGG16, and Xception) as well as an ensembled model. The models easily obtained excellent (99%) accuracy on the training/ validation sets, even without augmentation, though test set accuracy lagged behind (65-73%). Shifting images improved test set accuracy by 2-5%, depending on the model. Ensembling the three models together improved test set accuracy by another 5-11%, depending on the model. Augmenting images to increase and balance datasets, as well as ensembling well-performing models, holds great promise in improving machine learning (ML) models to classify various stages of AD.

# Contents

# 1 Introduction

Alzheimer's disease is the most common dementia, yet it remains difficult to diagnose. PET scans to measure amyloid protein build-up and cerebrospinal fluid (CSF) biomarkers can help to provide an accurate diagnosis, however they are expensive and invasive tests.1 MRIs are much more accessible.

However, MRIs depend on an experienced neurologist to interpret them. A dependable and accurate ML model has the potential to help a general practitioner in a small town to read an MRI as accurately as a specialist at a large research hospital. Even with a trained specialist, one study found experienced radiologists only had an accuracy of 57.5 to 70% percent when reading an MRI to diagnose AD.2 There has been progress in developing ML models for other conditions like breast cancer.3

In addition to diagnosing AD, being able to differentiate mild cognitive impairment from normal cognition is important for clinical trials, as potential treatments are more likely to be effective before the condition progresses to AD.

In the past, neurological ML models mainly used MRIs for feature extraction, i.e., to get a measurement for a part of the brain, such as the size of the hippocampus. While this has proved helpful, there is significant domain knowledge required in order to understand which parts of the brain to target. Extracting features is challenging; we also can't be sure that we are picking out the most helpful features.

CNNs offer a different approach of using computer vision to look at the image as a whole. There is less domain knowledge needed as the computer decides which are the relevant features. However, using a CNN network is not without its own challenges. CNN networks work best when you have lots of data and in dementia research there is a lack of labeled images. While some approaches have been explored using unlabeled data to help train a network4, the majority of researchers continue to use labeled datasets.

One way around a lack of data is data augmentation. In this paper, three different approaches are compared: using the original non-augmented dataset, using a flipped training set, and using a shifted training set.

Another method of dealing with a lack of data when creating a CNN model is to use transfer learning, i.e., using pretrained networks as a starting point for your model. There are plenty of other studies that have used these pretrained models to look at Alzheimer's imaging data,5 6 7 however relatively few that have looked at the benefits of ensembling models, which is the goal of this project.

# 2 Description of the Data Set

A Kaggle dataset was used for this project: Alzheimer's Dataset (4 class of images). This dataset contains 6400 jpg images (slices) taken from MRI images of possible Alzheimer's patients,

divided into 4 categories: non-impaired, very mild impairment, mild impairment, and moderate impairment. (See Figure 1)

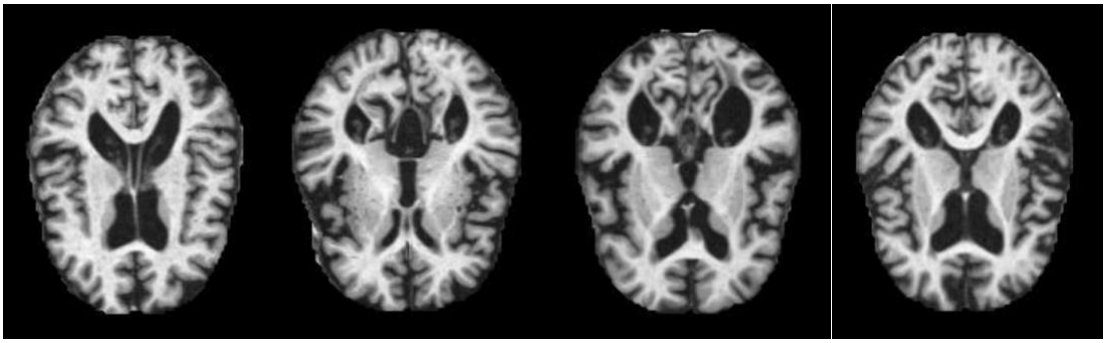


*Figure 1: The four classes of images. From left to right: non-impaired, very mild impairment, mild impairment, moderate impairment*

As you can see, the categories are not easily differentiable to the untrained eye, making this potentially a good task for machine learning. The original dataset was also quite unbalanced:

| **Train Set** | **# Images** |
|---|---|
| Non-impaired (cognitively normal) | 2560 |
| Very mild impairment | 1792 |
| Mild impairment | 717 |
| Moderate impairment | 52 |

| **Test Set** | **# Images** |
|---|---|
| Non-impaired (cognitively normal) | 640 |
| Very mild impairment | 448 |
| Mild impairment | 179 |
| Moderate impairment | 12 |

As people become more impaired, we have fewer images of their condition. As patients progress in an illness, it becomes harder for them to attend study appointments. Issues of consent also become more problematic as people progress into dementia.

# 3 Related Work/Algorithms Used

Most of the similar studies I found used only one pretrained network to classify the images, although they sometimes compared the results from more than one. Some of the pre-trained CNNs used were: VGG-195, ResNext1016 , GoogLeNet8, and Inception9.

Fang et al's 2020 study[10] is the most similar one I've come across to what I'm proposing. They also extracted 2D slices from the MRI and PET images, and then used three different pre-trained networks (GoogLeNet, ResNet, and DenseNet) to get a probabilistic score for each image.

Finally, they used a Decision Tree to combine the probabilistic scores from each CNN and get their final prediction.

The three pre-trained CNN networks I selected for this project are: ResNet50, VGG16, and Xception.

ResNet50 was the first CNN model to introduce the concept of a skip connection: a signal feeding into a layer is also added to the output of another layer that is located higher up in the stack. It won the ImageNet challenge in 2015. Input shape (224, 224, 3)

VGG16 won the ImageNet competition in 2014 and uses smaller filter, but a deeper network. It was named after the Visual Geometry Group from Oxford that developed it. Input shape (224, 224, 3)

Xception uses depthwise separable convolution and improves upon the Inception model. It dates from 2017 and is 71 layers deep. Input shape (299, 299, 3)

Finally, I used a standard random forest algorithm to ensemble these three different models together.

# 4 Experimental Setup

Because the original dataset was so imbalanced, I needed to figure out how I was going to augment the data. I decided to first divide the train set into train/validation, using a 70/30 split. I did this by moving 30% of the training images to a separate validation folder. The final non-augmented dataset consisted of:

| **Train Set** | **# Images** |
|---|---|
| Non-impaired (cognitively normal) | 1792 |
| Very mild impairment | 1254 |
| Mild impairment | 502 |
| Moderate impairment | 36 |

| **Validation Set** | **# Images** |
|---|---|
| Non-impaired (cognitively normal) | 768 |
| Very mild impairment | 538 |
| Mild impairment | 215 |
| Moderate impairment | 16 |

| **Test Set** | **# Images** |
|---|---|
| Non-impaired (cognitively normal) | 640 |
| Very mild impairment | 448 |
| Mild impairment | 179 |
| Moderate impairment | 12 |

I decided to perform the augmentation only on the *test* set. A large problem with CNN networks of MRI images is overfitting. You can train a network to identify images in one dataset, but often the model does not work on a different dataset. As the test set would, not be augmented, I did not augment the validation set as well, giving the model the best chance to train correctly.

**Flipping the Data**

First, I decided to augment the data by flipping the images. I used cv2.flip() to flip the jpgs in three directions: horizontally (using flip code 1), vertically (using flip code 0) and around both axes (using flip code -1). I also balanced the data for all the classes, except for the smallest minority class (I didn't have enough data to do that) by selecting varying numbers of the original images to flip. (See Figure 2)

| Flipped Train Set | # Images |
|---|---|
| Non-impaired (cognitively normal) | 2008 |
| Very mild impairment | 2008 |
| Mild impairment | 2008 |
| Moderate impairment | 144 |

The validation/test sets for the flipped data remained the same as for the non-augmented data.

**Shifting the Data**

I also wanted to try a transformation that would allow me to completely balance the dataset. I used cv2.warpAffine to slightly shift all the images. By using more shifts on the smallest class, I was able to balance the dataset. This seemed like it might be a good transformation to try because MRIs are all taken at slightly different angles, depending on the machine and the person. (See Figure 3)

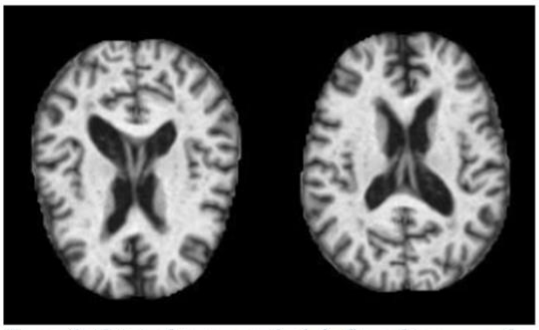


*Figure 2: Original image on the left; flipped image on the right*

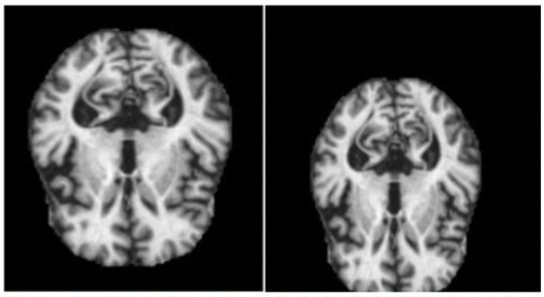


*Figure 3: Original image on the left; shifted image on the right*

| Shifted Train Set | # Images |
|---|---|
| Non-impaired (cognitively normal) | 2008 |
| Very mild impairment | 2008 |
| Mild impairment | 2008 |
| Moderate impairment | 2016 |

The validation/test sets for the shifted data remained the same as for the non-augmented data.

After augmenting the data, I had effectively created 3 different training sets: one with no augmentation, one with flipping the images, and one with shifted images.

Next, I ran each pre-trained model on each dataset, for a total of nine models. I then compared the results and selected the three best models to ensemble into a final model. See figure below to see how this was done:

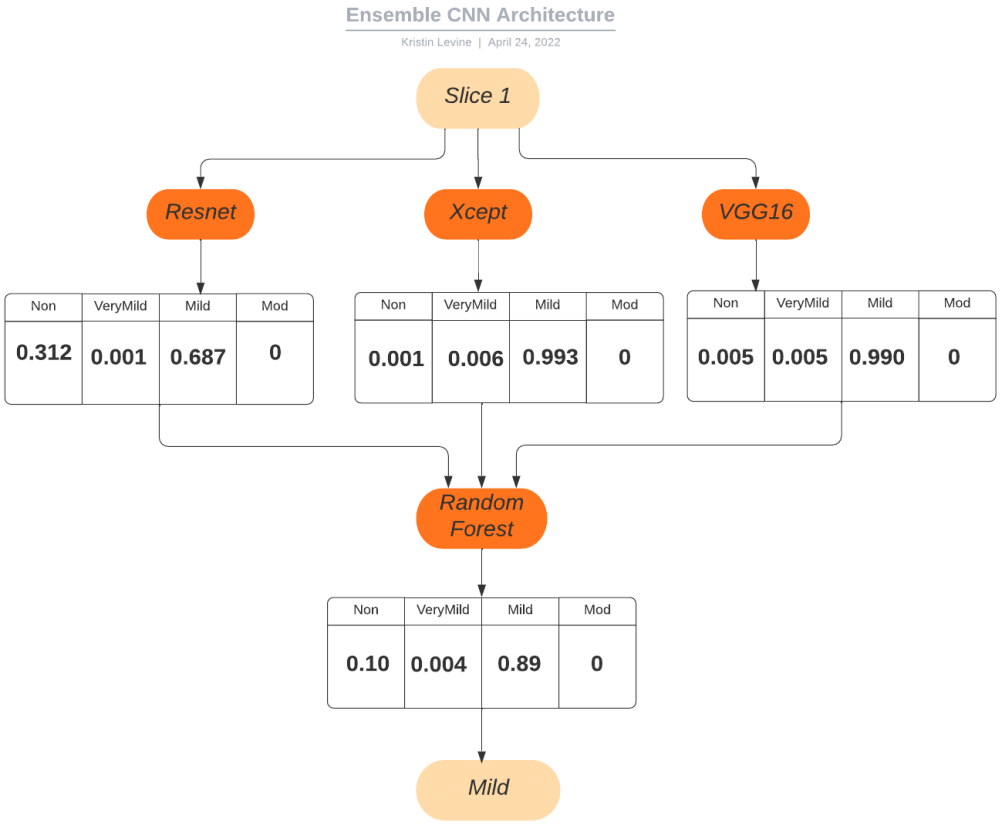



*Figure 4: For each slice, each model generates a probability for each class. These probabilities are combined in a data frame; a random forest is then run to come up with a final prediction.*

# 5 Results

## 5.1 Individual Model Results

I trained all the models on Google Colab Pro, using a batch size of 32, with dropout of 0.1. I trained the frozen layers for 10 epochs with a learning rate of 0.001, using both early stopping

and reduce LR on plateau callbacks. I then unfroze each model and trained it for an additional 10 epochs, starting with a learning rate of 0.0001. Here's a summary table of all my results:

| Model | Augmentation | Train Acc | Val Acc | Test Acc |
|---|---|---|---|---|
| ResNet | None | 1.00 | 0.9740 | 0.7303 |
| VGG16 | None | 0.8597 | 0.8699 | 0.6529 |
| Xception | None | 0.9980 | 0.9453 | 0.6701 |
| | | | | |
| ResNet | Flip | 0.9992 | 0.9818 | 0.7224 |
| VGG16 | Flip | 0.9162 | 0.8530 | 0.6482 |
| Xception | Flip | 0.9990 | 0.9655 | 0.7177 |
| | | | | |
| ResNet | Shift | 0.9981 | 0.9876 | 0.7482 |
| VGG16 | Shift | 0.9778 | 0.8783 | 0.6904 |
| Xception | Shift | 0.9994 | 0.9681 | 0.7209 |

All the models did quite well on the training and validation sets. The VGG16 model would probably have improved with more training, but I wanted to keep all the epochs the same in order to compare them. Here's a closer look at the individual model test set results for the best models – for all CNNs this was the shifted model.

**ResNet50**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| mild | 0.82 | 0.37 | 0.51 | 179 |
| mod | 1.00 | 1.00 | 1.00 | 12 |
| non | 0.77 | 0.86 | 0.81 | 640 |
| very | 0.70 | 0.73 | 0.72 | 448 |
| accuracy | | | 0.75 | 1279 |
| macro avg | 0.82 | 0.74 | 0.76 | 1279 |
| weighted avg | 0.75 | 0.75 | 0.74 | 1279 |

Accuracy : 74.8240813135262

**VGG16**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| mild | 0.58 | 0.37 | 0.45 | 179 |
| mod | 1.00 | 0.17 | 0.29 | 12 |
| non | 0.80 | 0.78 | 0.79 | 640 |
| very | 0.58 | 0.70 | 0.64 | 448 |
| accuracy | | | 0.69 | 1279 |
| macro avg | 0.74 | 0.50 | 0.54 | 1279 |
| weighted avg | 0.70 | 0.69 | 0.69 | 1279 |

Accuracy : 69.03831118060985

**Xception**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| mild | 0.80 | 0.50 | 0.62 | 179 |
| mod | 0.56 | 0.83 | 0.67 | 12 |
| non | 0.84 | 0.69 | 0.76 | 640 |
| very | 0.61 | 0.84 | 0.71 | 448 |
| accuracy | | | 0.72 | 1279 |
| macro avg | 0.70 | 0.72 | 0.69 | 1279 |
| weighted avg | 0.75 | 0.72 | 0.72 | 1279 |

Accuracy : 72.08756841282252

By ensembling the models, we can draw from their different strengths and hopefully minimize their weaknesses.

## 5.2 Ensembled Results

To ensemble the models, I took the individual CNN model with the best results – this was the shifted models for all three of the pre-trained models. I created a data frame of the results (see sample below), and then trained it again using a random forest, with a 70/30 train test split.

| | mild1 | mod1 | non1 | very1 | mild2 | mod2 | non2 | very2 | mild3 | mod3 | non3 | very3 | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 6.869561e-01 | 5.136539e-06 | 0.311814 | 0.001224 | 0.002607 | 4.616061e-08 | 0.589123 | 0.408270 | 0.999881 | 3.392238e-05 | 4.824037e-05 | 0.000036 | mild |
| 1 | 2.952032e-01 | 1.046946e-04 | 0.353254 | 0.351438 | 0.450478 | 5.934722e-07 | 0.050777 | 0.498744 | 0.877252 | 2.541937e-06 | 6.271919e-02 | 0.060026 | mild |
| 2 | 4.538712e-02 | 2.240127e-05 | 0.162286 | 0.792305 | 0.024907 | 8.539470e-08 | 0.089811 | 0.885282 | 0.743481 | 4.092175e-06 | 1.352151e-01 | 0.121299 | mild |
| 3 | 9.862058e-01 | 4.380234e-06 | 0.000948 | 0.012842 | 0.998428 | 8.421543e-08 | 0.000562 | 0.001010 | 0.996401 | 3.398987e-08 | 5.353014e-07 | 0.003598 | mild |
| 4 | 8.418527e-01 | 4.924598e-07 | 0.046375 | 0.111772 | 0.647946 | 5.496316e-06 | 0.012049 | 0.339999 | 0.824395 | 9.852059e-07 | 6.257508e-03 | 0.169346 | mild |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1274 | 1.225779e-05 | 5.469460e-05 | 0.181109 | 0.818824 | 0.059487 | 4.190398e-06 | 0.242797 | 0.697712 | 0.001214 | 1.274106e-01 | 5.462531e-03 | 0.865913 | very |
| 1275 | 2.331303e-05 | 1.243776e-06 | 0.066117 | 0.933858 | 0.000153 | 2.476680e-09 | 0.037964 | 0.961883 | 0.122486 | 4.676188e-04 | 1.695602e-03 | 0.875350 | very |

Here are the final results of that model.

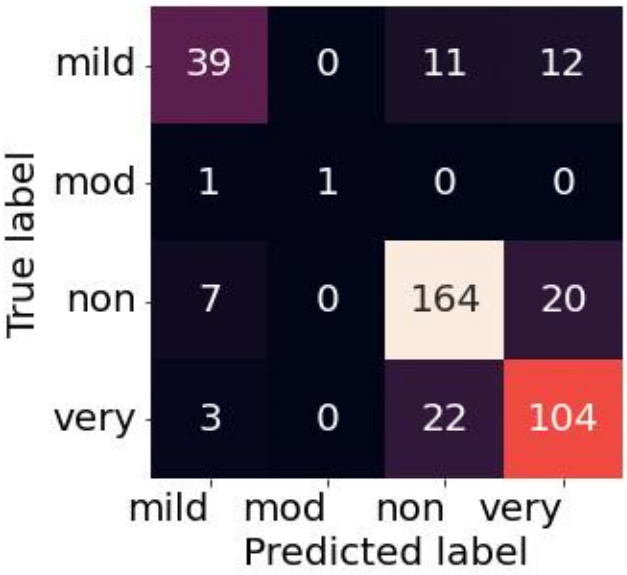
Results Using All Features:

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| mild | 0.78 | 0.63 | 0.70 | 62 |
| mod | 1.00 | 0.50 | 0.67 | 2 |
| non | 0.83 | 0.86 | 0.85 | 191 |
| very | 0.76 | 0.81 | 0.78 | 129 |
| accuracy | | | 0.80 | 384 |
| macro avg | 0.84 | 0.70 | 0.75 | 384 |
| weighted avg | 0.80 | 0.80 | 0.80 | 384 |

Accuracy : 80.20833333333334

The confusion matrix below provides another view of the results.




# 6 Discussion

As we can see each of the pretrained models did an excellent job on the training and validations sets; all of them struggled to deal with the test set.

It is interesting to note that flipping the images did *not* help our models; in fact, for two of our models, ResNet and VGG16, it made it worse. This is probably due to the fact that all these images were already standardized to face in one direction. It is also unclear if you can flip images of the brain – different hemispheres play different roles. It is possible that flipping the images actually confuses the models, because it now sees changes in a different part of the brain.

Shifting our images, however, was helpful. It improved our test accuracy for all three of our models – improving our ResNet model by about 2%, the VGG16 model by about 4%, and the Xception model by 5%.

| **Model** | **Test Acc (No Aug)** | **Test Acc (Shifted)** | **Improvement** |
|---|---|---|---|
| ResNet | 0.7303 | 0.7482 | 0.0179 |
| VGG16 | 0.6529 | 0.6904 | 0.0375 |
| Xception | 0.6701 | 0.7209 | 0.0508 |

We can see that the models also varied in how well they classified each category.

| Class | Precision | | | Recall | | | F1-score | | |
|---|---|---|---|---|---|---|---|---|---|
| | R | V | X | R | V | X | R | V | X |
| Mild | 0.82 | 0.58 | 0.80 | 0.37 | 0.37 | 0.50 | 0.51 | 0.45 | 0.62 |
| Mod | 1.00 | 1.00 | 0.56 | 1.00 | 0.17 | 0.83 | 1.00 | 0.29 | 0.67 |
| Non | 0.77 | 0.80 | 0.84 | 0.86 | 0.78 | 0.69 | 0.81 | 0.79 | 0.76 |
| Very | 0.70 | 0.58 | 0.61 | 0.73 | 0.70 | 0.84 | 0.72 | 0.64 | 0.71 |

For the “mild” class:

- ResNet had the best precision (0.82)
- Xception had the best recall (0.50) and f1 score (0.62)

For the “mod” class:

- ResNet had perfect precision, recall, and f1-score for this class (note: the test set was very small – 12 images)

For the “non” class:

- Xception had the best precision (0.84)
- ResNet had the best recall (0.86) and f1 score (0.81)

For the “very” class:

- ResNet had the best precision (0.70) and f1 score (0.72)
- Xception had the best recall (0.84)
-

Ensembling the models results in a dramatic improvement in our accuracy for the test set data, ranging from 5-11%.

| Model | Test Acc (Shifted) | Test Acc (Ensembled) | Improvement |
|---|---|---|---|
| ResNet | 0.7482 | 0.8021 | 0.0539 |
| VGG16 | 0.6904 | 0.8021 | 0.1117 |
| Xception | 0.7209 | 0.8021 | 0.0812 |

After seeing that the VGG16 model did not perform the best in any of the categories, I reran the ensembled model using only the two highest models: Shifted ResNet (0.7482) and Shifted Xception (0.7209). It is interesting to note that this two-model ensemble did not perform as well, achieving a test accuracy of only 0.7839. So while the VGG16 model may not be the best in any one category, adding it to the ensembled model improved the combined model accuracy by almost 2%.

# 7 Conclusion

When comparing non-augmented, flipped, and shifted datasets, shifting the images to create more data improved test set accuracy – even as it had no effect on the accuracy of train and validation sets. In this study, it improved test performance by 2-5%, depending on the model used.

In a field such as Alzheimer's research, shifting images to create more data is a way to improve test-set performance and perhaps increase the transferability of the model to other datasets.

Even when various models provide good results individually on train and validation sets, performance can be increased on the test set by ensembling them together to take advantage of their individual strengths and minimize their weaknesses. In this study, ensembling three models improved test performance by 5-11%. This benefit was seen even when all of the models had excellent accuracy on the training and validation sets; it even improved our final score when one of the models was a bit weaker than the other two.

Ensembling is another method that could be put to good use when dealing with small, limited datasets in neurodegenerative disease research.

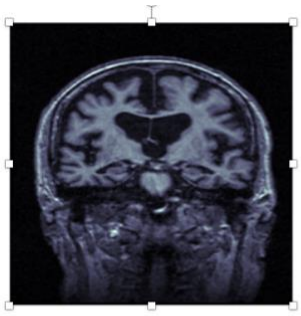
Further research directions might include more detained investigations into the optimal amount to shift images; exploring more pre-trained models; and/or investigating other image transformations such as brightness or normalization techniques.

# 8 Attempts at Creating my Own Dataset

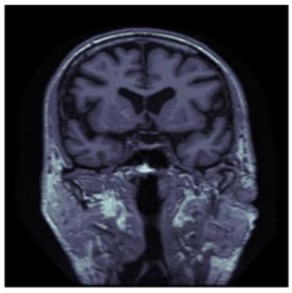
There was very little information about this dataset on Kaggle – no info on where it came from, how the patients were diagnosed, etc. It also contained no genetic or clinic data. I wanted to see if I could create my own dataset to give me more control over these variables.

I applied for and received access to Alzheimer's Disease Neuroimaging Initiative (ADNI), a public-private collaboration studying Alzheimer's disease. The initiative has been around since 2004 [11] and collects clinical, cognitive, imaging, biomarker, and genetic data to share with researchers. Through ADNI I was able to access original MRI images in the NiFTI format.

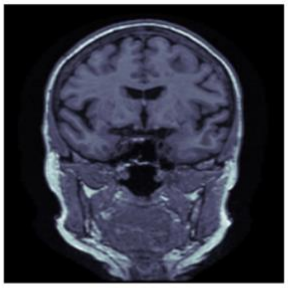
I started by downloading original NiFTI files from ADNI that fell into three different categories: cognitively normal (CN), mild cognitive impairment (MCI), and Alzheimer's disease (AD). Here are three sample images:



Alzheimer's Disease (AD)



Cognitively Normal (CN)



Mild Cognitive Impairment (MCI)

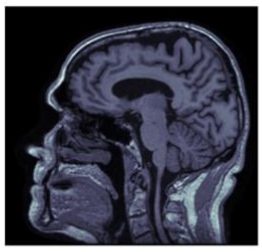
Using a Python package called nibabel I was able to view these 3D images, as well as take slices from each of the different orientations.

```
nifti = nib.load("137_S_0796.nii")
print(nifti.shape)
```

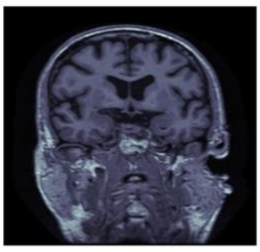
```
(256, 256, 180)
```

```
nii = nib.load(nii_path).get_fdata()[:,:,90] #sagital
nii = nib.load(nii_path).get_fdata()[:,128,:] #coronal
nii = nib.load(nii_path).get_fdata()[128,:,:] #axial
```

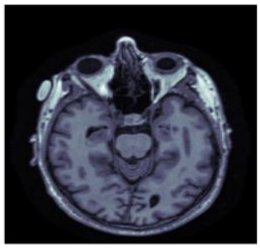
Here is an example of slices from each of the three different orientations:



Sagittal View



Coronal View



Axial View

You may notice that these images look different than the images used in the Kaggle dataset. These are the raw images that have not yet had the "skull stripping" done – a complicated process, usually using the FreeSurfer software.

You may also notice that all of the images in the Kaggle dataset are from the same orientation. My idea was to extract a set of slices from these images from different orientations, and then try a similar set of pretrained models on this new set of slices. I wanted to try skipping the preprocessing step using FreeSurfer to remove the skull and other image artifacts, hypothesizing that perhaps this was not necessary for the CNN models to accurately classify the images.

This turned out not to be true. While I was able to extract the slices (I did 21 in each orientation) and I got good train/and validation accuracy, the test accuracy refused to budge, hovering right about 40%, which with only three classes, wasn't much better than chance. The models appeared to be drastically overfitting to my test images.

| **Model** | **Train Acc** | **Val Acc** | **Test Acc** |
|---|---|---|---|
| ResNet | 0.9999 | 0.9982 | 0.4018 |
| VGG16 | 0.9741 | 0.9667 | 0.3993 |
| Xception | 0.9999 | 0.9977 | 0.3900 |

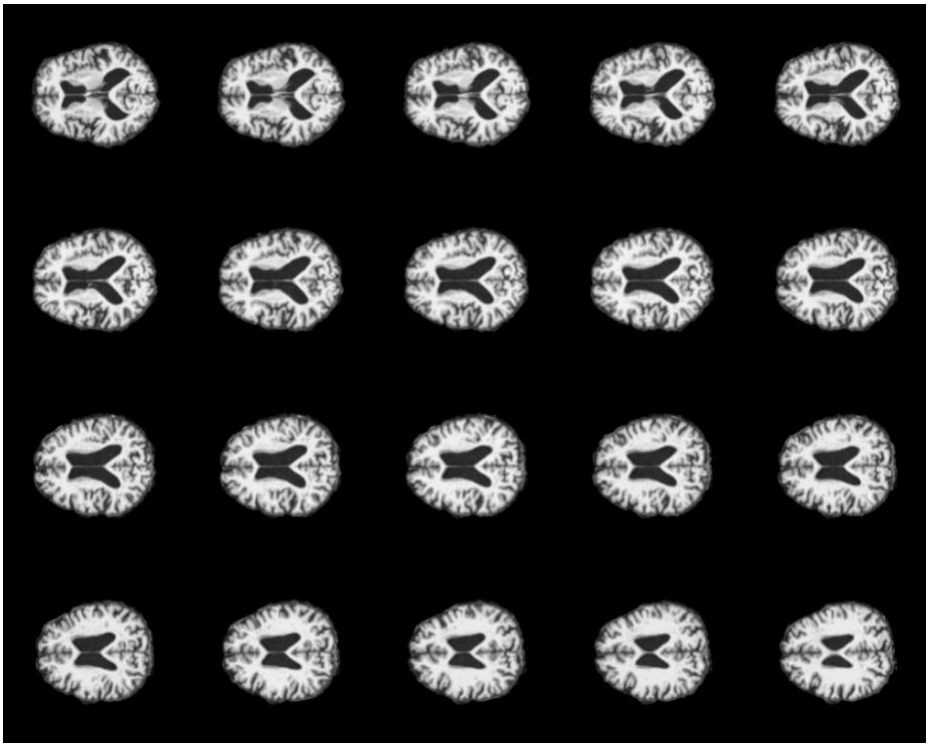
I realized that the FreeSurfer preprocessing did appear to be necessary – without the preprocessing there was just too much extra/confusing data. I liken it to looking for a needle in a haystack. If you know the needle (the information we need) is in the bottom half of the haystack, it will make your search faster and more accurate if you remove the top half of the hay. A brain disorder is unlikely to be seen in the bones of the skull; therefore, if we remove that info, we can actually create a better model.

Luckily, in addition to raw MRI images, ADNI also contains NiFTI images that have been preprocessed (skull stripped) with the FreeSurfer software. I downloaded some of these images as well.

After studying the images from the Kaggle dataset, I decided to take images that visually looked similar and were from the same orientation. In manually exploring 10 images from each class, I discovered another challenge when working with MRI images – the relevant slices vary from person to person.

For example, for one patient, the code for slices 160-180 extracted the slices I thought had the best chance of containing relevant information.

```
image = nib.load(nii_path).get_fdata()[:,:,160:180, 0]
```



However, for another patient, the range of slices was 135:155.

```
image = nib.load(nii_path).get_fdata()[:,:,135:155, 0]
```

If I only took 21 slices in an orientation, I might miss most (if not all) of the relevant info.

Other researchers have dealt with this problem in different ways. Some took a standard range of slices9; others selected only the median slice [12].

The other problem I was hoping to investigate was the problem of data leakage.[13] For example, if you take 2000 MRIs and take 50 slices from each one, you'd end up with 100,000 images. If you then divide them into train/val/test sets, images from the same person may be in the train and test set. These images may be very similar – in effect, the test set might contain images the model had already seen. Since I was unsure of how the Kaggle dataset was created, I could not be sure it did not suffer from data leakage issues.

To avoid this problem, you need to divide the images before you take the slices. However, if you are not careful in your data preprocessing, you again risk having the same patient in the train/test group, because many patients come back for more than one brain imaging session.

Finally, there is the issue of taking multiple slices from each person, but making a per-slice prediction. Some researchers have argued it makes much more sense to make a per-patient prediction.[13] This might also help to deal with the overfitting common with these types of models, as each model could be less-accurate (or fitted to a particular dataset), as long as the combined-slice prediction had good accuracy.

I attempted this type of prediction – taking 21 slices from each orientation and then averaging all 63 slides together to make one final prediction.

| | AD | CN | MCI | Prediction | target |
|---|---|---|---|---|---|
| **3402** | 9.829723e-15 | 0.755327 | 0.244673 | 1 | 2 |
| **3403** | 9.320969e-12 | 0.416101 | 0.583899 | 2 | 2 |
| **3404** | 1.108769e-11 | 0.484034 | 0.515966 | 2 | 2 |
| **3405** | 3.003239e-13 | 0.630967 | 0.369033 | 1 | 2 |
| **3406** | 3.435860e-17 | 0.971632 | 0.028368 | 1 | 2 |
| ... | ... | ... | ... | ... | ... |
| **3460** | 1.762819e-01 | 0.111977 | 0.711742 | 2 | 2 |
| **3461** | 1.018578e-01 | 0.123774 | 0.774368 | 2 | 2 |
| **3462** | 1.139304e-02 | 0.151275 | 0.837332 | 2 | 2 |
| **3463** | 1.746368e-02 | 0.146033 | 0.836503 | 2 | 2 |
| **3464** | 2.256335e-03 | 0.165082 | 0.832662 | 2 | 2 |

63 rows × 5 columns

Final Prediction

| | Subject | AD | CD | MCI | Prediction | Target |
|---|---|---|---|---|---|---|
| **0** | 55 | 0.06 | 0.4 | 0.54 | MCI | MCI |

While this model worked for this particular patient, my accuracy continued to hover at only 40% -- not much better at chance. It seems there is some knack to selecting the slices from each MRI image that I have not yet figured out.

However, while this additional part of my project didn't work out quite like I expected, I learned so much. First of all, I learned how to load and slice original MRI images. I gained a greater understanding of how complex MRI images are. Different machines have different settings. People move during scans. All these variables have to be taken into account and adjusted for.

Second, I realized how important domain knowledge is for this task. For example, in Alzheimer's disease the hippocampus is one of the parts of the brain that is earliest to show damage. A neurologist – knowing exactly where this is and which parts to target – would be invaluable in designing a diagnostic model and determining which brain slides contain the most useful information.

Finally, having a "tolerance for failure" on this project allowed me to explore some of the limits of CNNs. They aren't magic, and with supervised learning, are truly as only as good as your dataset. I see now why creating such a dataset is such a challenge. In addition to all the technical aspects, neurodegenerative diseases are a special challenge because people can shift from one category to another, unlike a dog or a cat who always remains one or the other. For example, an image may be labeled "CN" at a first visit with a physician; however, at a follow up visit perhaps a year or two later, an image from the same patient may be labeled "MCI."

It is unclear then, which is the "correct" label for that patient. It is possible that the brain structures actually changed from one visit to the next, so both labels are "correct." However, it is also possible that signs of MCI were actually in the first MRI but were so subtle they were missed by the person doing the annotating. Unsupervised learning techniques to help group images correctly may be helpful in this regard.

These challenges are also why these types of models are worth pursuing. If we know which people with CN are more likely to move to MCI or MCI to AD, we'd have a better idea of who would be more likely to benefit from potential therapies and treatments. Perhaps the CNN models will be able to see signs of coming changes before human doctors can.

So while robots aren't going to be reading our brain MRIs tomorrow, they just might be assisting neurologists sometime soon.

**Data and Code Availability**

The datasets analyzed during the current study are available here. Code notebooks are available here.

# 9 Bibliography


1. Sun, H., Wang, A., Wang, W. & Liu, C. An Improved Deep Residual Network Prediction Model for the Early Diagnosis of Alzheimer's Disease. *Sensors* vol. 21 4182 (2021).
2. Syaifullah, A. H. *et al.* Machine Learning for Diagnosis of AD and Prediction of MCI Progression From Brain MRI Using Brain Anatomical Analysis Using Diffeomorphic Deformation. *Front. Neurol.* **11**, 576029 (2020).

3. Yala, A. *et al.* Multi-Institutional Validation of a Mammography-Based Breast Cancer Risk Model. *J. Clin. Orthod.* JCO.21.01337 (2021).
4. Alzubaidi, L. *et al.* Novel Transfer Learning Approach for Medical Imaging with Limited Labeled Data. *Cancers* **13**, (2021).
5. Helaly, H. A., Badawy, M. & Haikal, A. Y. Deep Learning Approach for Early Detection of Alzheimer's Disease. *Cognit. Comput.* 1–17 (2021).
6. Li, Y. *et al.* Transfer learning-trained convolutional neural networks identify novel MRI biomarkers of Alzheimer's disease progression. *Alzheimers. Dement.* **13**, e12140 (2021).
7. Tanveer, M. *et al.* Classification of Alzheimer's disease using ensemble of deep neural networks trained through transfer learning. *IEEE J Biomed Health Inform* **PP**, (2021).
8. Prakash, D. *et al.* A Comparative Study of Alzheimer's Disease Classification using Multiple Transfer Learning Models. *Journal of Multimedia Information System* 6, 209–216 (2019).
9. Tufail, A. B., Ma, Y.-K. & Zhang, Q.-N. Binary Classification of Alzheimer's Disease Using sMRI Imaging Modality and Deep Learning. *J. Digit. Imaging* **33**, 1073–1090 (2020).
10. Fang, X., Liu, Z. & Xu, M. Ensemble of deep convolutional neural networks based multi-modality images for Alzheimer's disease diagnosis. *IET Image Proc.* **14**, 318–326 (2020).
11. Wyman, B. T. *et al.* Standardization of analysis sets for reporting results from ADNI MRI data. *Alzheimers. Dement.* 9, 332–337 (2013).
12. Valliani, A. & Soni, A. Deep Residual Nets for Improved Alzheimer's Diagnosis. in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics* 615 (Association for Computing Machinery, 2017).
13. Yagis, E. *et al.* Effect of data leakage in brain MRI classification using 2D convolutional neural networks. *Sci. Rep.* **11**, 22544 (2021).