IMPROVED MEDICAL IMAGE CLASSIFICATION THROUGH TEST-BASED LEARNING

Sushmitha Kudari, Eron Johnson, Ali Zaidi, and Bryce Smith

*University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

In the field of image processing, training a model to classify images is well understood. The task increases in difficulty with the number of classes and diversity of the dataset. Trained models are often better at recognizing certain classes than others. This is most often due to class-similarity, which means that one class is very similar to another. Intuitively we see that the more similar pictures are the more difficult it is to distinguish between them. In solving this problem we turn to a technique for teaching humans; learning by passing tests. It is common practice to evaluate a person's understanding of materials by giving them tests of increasing complexity over an array of topics to see where their aptitudes and weaknesses lie. Once weaknesses are discovered the person can study and improve in those areas to be able to pass the next test. In this way, training can be done more quickly by focusing on weaknesses rather than retraining on information which is already known. We plan to implement this methodology to create a more robust and uniformly accurate classification model over a shorter training period.

Index Terms— Computer Vision, Convolutional Neural Nets, Multi-variable Classification

1. IMPORTANCE OF THIS PROBLEM

When developing computer vision models, recognizing specific objects or states is currently the focus of intense research: Accurately identifying cars and traffic signals is fundamental to self driving cars, face and body recognition is important for security and health-policy research, there is seemingly no end to potential applications. The main obstacle to quick and accurate classification in computer vision is the diversity and complexity of most images. Most models require extensive training to perform accurate classification which means that training time is the single most costly factor in developing any model. If our methodology allows for reliably faster training of models it would prove to be an extremely valuable technique for machine learning. We have decided to test our model on a dataset of MRI images used to classify stages of Alzheimer's disease (AD).

To ECE 269, Professor Pengtao Xie and our wonderful TA's.

1.1. Applications for Alzheimer's Disease

Alzheimer's disease is a chronic neurodegenerative disease which develops slowly and then worsens over time. The risk of developing Alzheimer's increases with age and the number of people who will develop Alzheimer's disease is expected to skyrocket over the next few years as seventy-nine million adults reach retirement [6]. Early diagnosis of Alzheimer's gives people the best chance for being treated effectively [6], which makes detecting and classifying the stage of AD in a patient is vital to their treatment. The most accurate method for classifying Alzheimer's is by identifying structural changes in the brain through magnetic resonance imaging (MRI). It has been demonstrated that this image-based classification is a task well suited for modern computer vision, and there are emerging implementations which assist doctors in making diagnosis. We plan to show that our model can match or outperform current implementations in classifying four distinct stages of AD, and learn to do so in a shorter training period. The primary method of diagnosis is analysis of MRI images of the patient's brain. Our approach implements a VGG16 architecture trained on a series of these MRI images.

2. EXISTING WORKS

2.1. Current Applications

As early as 1999 [7], researchers have been employing various machine learning methods to detect AD onset from MRI data alone[8]. Researchers began by establishing biomarkers to be evaluated. A biomarker is a quantifiable indicator of a biological state. Hippocampal volume has been a key biomarker that doctors have used in the past, and it has shown high diagnostic accuracy for AD. However, hippocampal volume alone is considered clinically insufficient as a predictor of progression from mild cognitive impairment (MCI) to AD[9]. Hence there is a need to look at more features and recent research has shown that important changes in many other cortical regions also occur[10]. The spatial features in MRI scans make them ideal for machine learning and deep learning methods which are good at aggregating such features for classification.

Plant et al.[11] used feature selection to achieve 92% accuracy in a binary classification of AD. Their paper demonstrates that principal component analysis, independent component analysis, structural equation modeling, and support vector machines have recently shown promise in analyzing structural MRI to detect spatial patterns of atrophy associated with AD.

Wang et al.[12] used DenseNet and ensemble methods to classify the entire 3D MRI scan leading to a new state of the art three class classification (Alzheimer versus Mild Cognitive Impairment versus Cognitively Normal) with an accuracy of 97%. They employed a Deep Neural Network containing autoencoders to combine features from regions of interest (ROI) in PET and MRI scans. ROI-based methods such as this can extract representative features and partly reduce the feature dimension, but risk being too empirical as they are based on qualitative observation. This is not sufficient to capture the more granular features associated with AD. Additional methods proposed would capture successive slices from MRI scans, because it was deemed that each slice covers significant areas for dementia detection.

With 3D MRI there have also been developments in applying 3D convolution networks. One was combining Sparse AutoEncoders with 3D CNN and perform inference on complete MRI scans. Another researcher[11] working with our same dataset extracted a number of 3D patches from the whole MRI and a patch transformed into features by 3D-CNN. Finally, multiple 3D CNNs were used to combine the features and classify them.

2.2. Prior Research

The methodology used to train our tester model is based on Learning by Passing Tests, with Application to Neural Architecture Search [1]. The learner model will be trained using increasingly difficult images provided by the tester model that we create. This will essentially replicate human behavior in that a model cannot receive a more difficult image without learning to pass a less difficult task. The tests will focus on areas in which the learner shows poor performance.

2.3. Dataset

We obtained our dataset from an online Kaggle challenge of MRI brain images[3]. The images are black and white with a standard dimension of 176 x 208 pixels. As given, there are about 5000 training images and 1200 test images. The dataset is separated into 4 classes: Non-Demented, Very Mildly Demented, Mildly Demented, and Moderately Demented. Each image is a 2D MRI cross section of the brain as shown in Figure 1. The images are planes taken at different heights of the brain. We chose this dataset for its reasonable size and consistency. All images came with the brain cross-section centered in the image and with the background cropped out. A disadvantage of this dataset was that it came with little information regarding the composition of the testing and training sets.

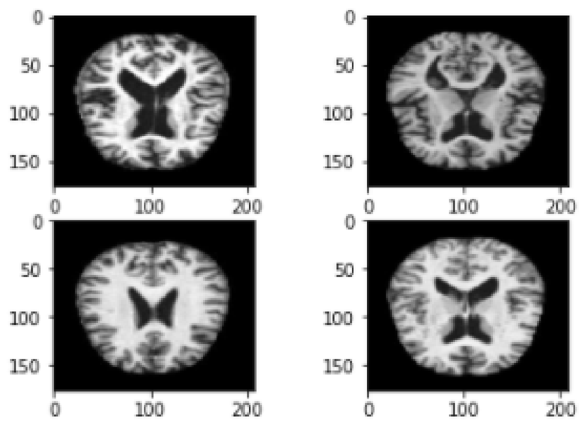

Fig. 1: MRI Images

3. ALGORITHM

3.1. Learner Model

For the learner we considered several models such as DenseNet and Inception, but ultimately went with VGG16 because we decided the model architecture is robust and also simpler to debug. VGG16 is widely known as one of the first neural networks to perform well at the Large-Scale Visual Recognition Challenge using a deep network.

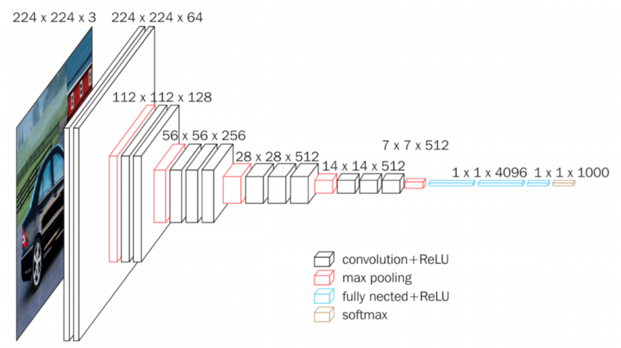

Fig. 2: VGG16 Architecture

VGG16 is a standard CNN which consists of layers of convolutional blocks with max pooling layers in between and several fully connected layers at the end of the network. The last fully connected layer has the same number of nodes as the number of classifiers with the SoftMax activation function. The SoftMax function is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \ for\ i = 1, ..., K\ and\ z = (z_1, ..., z_K) \in R^K \quad (1)$$

This formula computes the conditional probability of each class given the input (image) to the model. There is a conditional probability calculated for each of the classifications. The model predicts the image classification by picking the class with the highest conditional probability. We modified the original VGG16 network by inserting new fully connected layers at the end of the network that were untrained. We then retrained the entire network using our training set. For our loss function we used categorical cross entropy as it is widely used for multiclass classification problems. This method compares the distribution of the predictions (the activations in the output layer, one for each class) with the true distribution, where the probability of the true class is set to 1 and set to 0 for the other classes. Thus the true class is represented as a one-hot encoded vector and the closer the model's outputs are to that vector the lower the loss. The equation for this is defined as:

$$H(p,q) = -\sum_{x} p(x)\, log(q(x)) \quad (2)$$

We had originally planned to use an Adam optimizer which is an adaptive moment optimizer. This has the benefits of being straightforward to implement, computationally efficient, and its hyper-parameters have intuitive interpretation typically requiring little tuning. However, we found through research that an Adam optimizer might not be well suited for our needs as they function best for data which has noise, sparse gradients, non-stationary objectives, and non-convex optimization problems. Our classification problem has stationary targets and a generally convex solution.

Our research indicated that stochastic gradient descent (SGD) with Nesterov momentum is extremely well suited for image classification, particularly medical images. Since our function is relatively smooth this iterative form of optimization was extremely effective. The Nesterov momentum helped the optimizer to move through local minima which may be encountered on the way to the desired weights. We saw an immediate improvement of several percent in initial test accuracy when we switched from Adam to SGD with Nesterov. We can see in Figure 3 that after thirty epochs of training the model begins to plateau as its hidden units reach correct weights. The role of hidden units in neural networks is to approximate a 'function' efficiently from the available data-samples which can be generalized to unseen data. The problem is that the available data often has underlying nonlinear patterns which can only be extracted by using a nonlinear approximation function. So when the hidden units try to approximate a function for these samples they tend to fit a higher order approximator (they try to create a function that accurately approximates each instance in the training set) and in doing so they overfit to the data-samples. Overfitting causes a model to perform well on a training set but be unable to recognize similar patterns in new data. Dropout layers were added to the network model for this reason. The drop out technique simply turns off some of the hidden units in previous layers essentially 'dropping out' some of the parameters during the learning process. This is done so that models do not learn every redundant detail in the training set which ultimately helps prevent overfitting the model on the training set. The ideal model only learns the most efficient representations of the dataset that also apply to new scenarios.

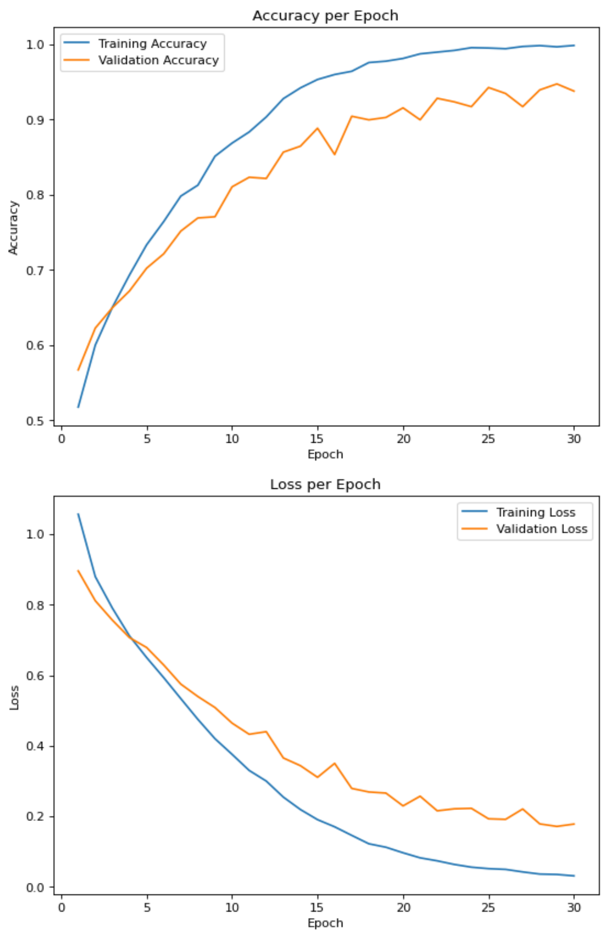

Fig. 3: Model Accuracy as it Trains

3.2. Tester Model

For our tester model, we elected to use a Support Vector Machine (SVM) in tandem with a Confusion matrix from the learner results. Upon completing a round of testing from the learner model, a Confusion matrix is generated and passed to the tester model as seen in Figure 4. The tester views which classes the learner is struggling with and creates tests which are more heavily representative of those classes. Classes along the diagonal of the confusion matrix are correctly identified, and values off the diagonal represent an instance where the prediction did not match the ground truth. We take the highest values of the off diagonal terms to determine which two classes the model is struggling to correctly classify and put images from those classes into our SVM model. We use the SVM to give us a mathematical framework to help distinguish between easy and hard images. For example; if

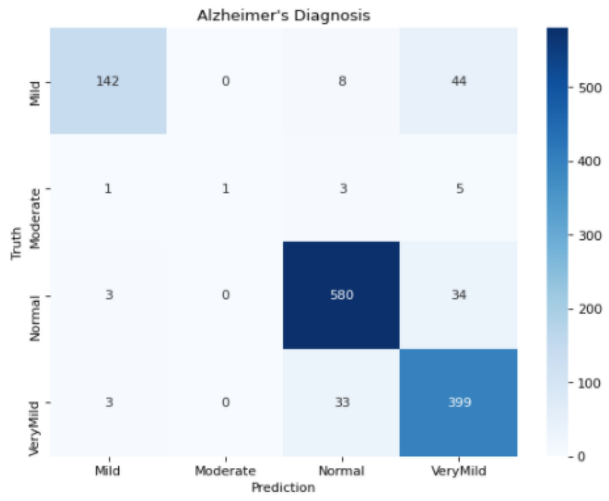

Fig. 4: Confusion Matrix of All Classes

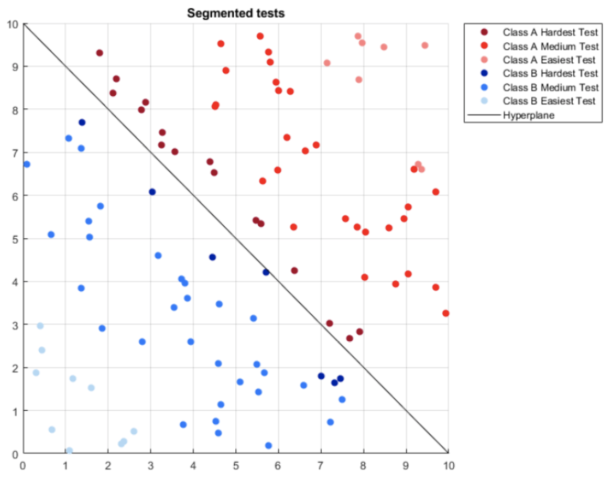

Fig. 5: SVM Generated From Two Classes

the Confusion matrix showed the model struggling to distinguish between Moderate Demented and Mild Demented Alzheimer's MRI images, an SVM with Moderate Demented and Mild Demented images will be generated and the proximity of images to the hyperplane will be used to help create tests with as seen in Figure 5. Simultaneously the learner is trained on more tailored sets based on these test results. Then we create progressively harder tests with images closer to the hyperplane.

3.3. SVM

The support vector machine we are built to find the hyperplane between two data classes of interest follows the pseudocode outline below:

```
def hyperplane(data,class_labels):
    model = SVC(kernel='linear')
    model.fit(data,class_labels)
    return model.decision_function()

def determine_image_difficulty(image):
    return min(euclidean distance from the hyperplane)

def assign_test(current_difficulty, pass_or_fail)
    if pass
        # if image passes the classification
        # give it a more difficult image
        while (determine_image_difficulty(select_random_image)
                < current_difficulty ):
            Select_random_image = select_random_image
        # set an iteration limit so program does
        # not run forever
        if number_of_iterations ¿ image_bank_size
            # indicator that this is the best the model can do
            return "BEST"
        return select_random_image
    else:
        # image can exist in either of the
        # two classification sets
        return random_closest_image to current_difficulty
```

3.4. Complete Model Architecture

Figure 7 below is a flow chart of our entire model intended to illustrate the tasks of the tester and iterative characteristics of our solution to learning by passing tests. Following the example of Du and Xie [1] our proposed solution to image recognition machine learning by passing tests went as follows:

1. A test bank of all images from the brain MRI dataset is provided to the tester model.

2. A randomized subset from the test bank is partitioned as the initial training and testing set to establish a baseline confusion matrix. This dataset is divided evenly into a training set and a validation set. The training set is then used as the training data of the learner. The validation set is used to evaluate the performance of the learner, which the tester model will build a baseline confusion matrix from.

3. The tester identifies the two lowest performing classes from the initial model and directs the SVM to find a hyperplane between the two classes of choice.

4. Using the data map and location of the hyperplane provided by the SVM, the tester selects a training and validation set from each class.

5. The validation set is further separated for the tester model into subsets by their score of difficulty based on their normalized distance from the hyperplane margins.

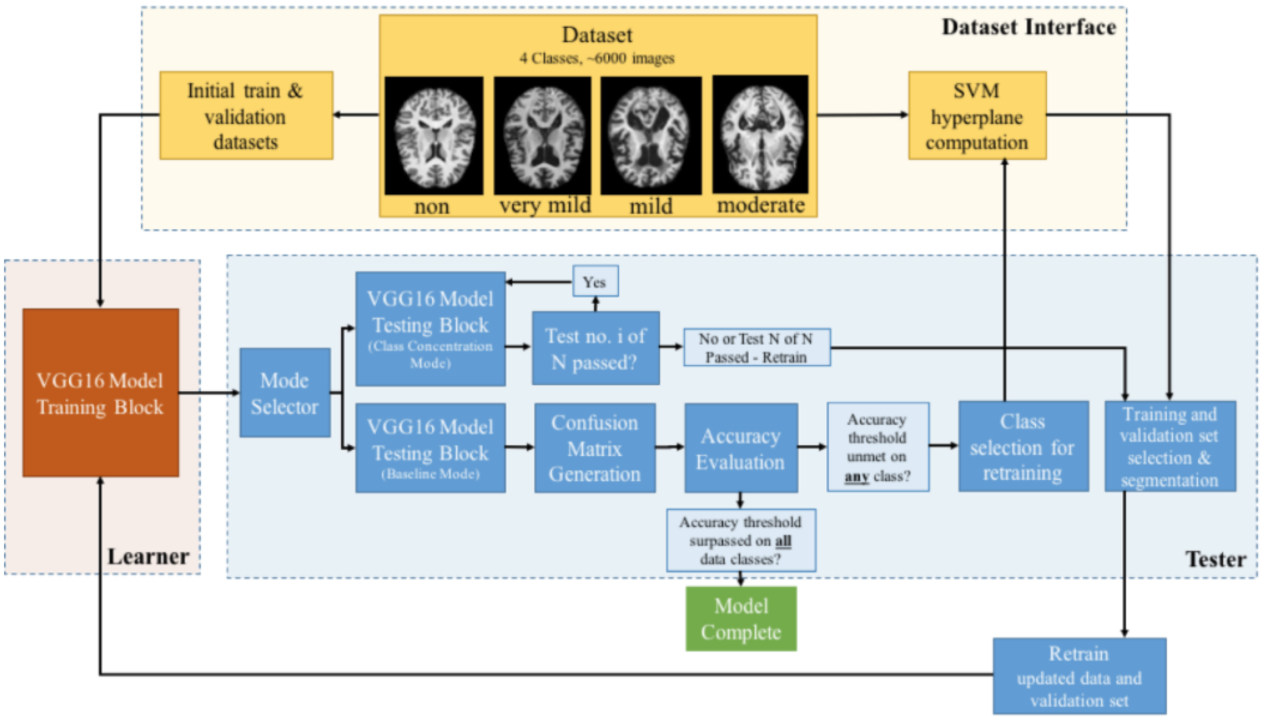
Dataset Interface
Dataset
4 Classes, ~6000 images
non
very mild
mild
moderate
Initial train & validation datasets
SVM hyperplane computation
VGG16 Model Training Block
Learner
Mode Selector
VGG16 Model Testing Block (Class Concentration Mode)
Yes
Test no. i of N passed?
No or Test N of N Passed - Retrain
VGG16 Model Testing Block (Baseline Mode)
Confusion Matrix Generation
Accuracy Evaluation
Accuracy threshold unmet on any class?
Class selection for retraining
Training and validation set selection & segmentation
Accuracy threshold surpassed on all data classes?
Tester
Model Complete
Retrain updated data and validation set

Fig. 6: Modified LPT Algorithm

6. The tester model supplies a portion of the training set to the learner model to refine the baseline model from.

7. When training is complete, the learner is evaluated on its performance of recognition of the simplest validation set of the two challenging classes.

8. If the accuracy does not meet a predetermined threshold value, steps 6-7 are repeated with a new training set.

9. If the accuracy meets a predetermined threshold value, more challenging tests are introduced to the learner model until a satisfactory accuracy is achieved for the most difficult tests from the identified classes.

10. A test similar to the initial training set with a more even distribution of classes from the dataset is then administered to the model to build a new confusion matrix.

11. The process in steps 3-10 are repeated until the learner performs with the goal accuracy on all classes in the dataset.

4. MATHEMATICAL ARCHITECTURE

A basic linear SVM model as shown in Figures 5 and 6 is used to classify the difficulty of images. We can assume two distinct classes of Alzheimer's MRI, and the line separating the two classes is called the hyperplane. While the image

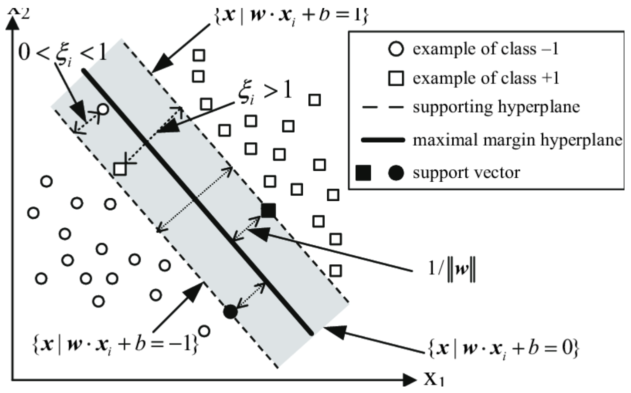

Fig. 7: Support Vector Machine

is represented in 2-dimensions above, it is in actuality a 3-dimensional space where the hyperplane is a 2 dimensional element.

The hyperplanes are the deciding factor in determining the difficulty of an image. The closer an image is to a hyperplane, the more difficult of a time the model will have to classify it. The hyper plane is defined using the mathematical model shown where $w * x + b$ is the equation of the hyperplane.

$$h(x_i) = \begin{cases} +1 & if\ w \cdot x + b \geq 0 \\ -1 & if\ w \cdot x + b < 0 \end{cases}$$

The data above the hyperplane (late stage) is classified as +1 while the data below the line (mild stage) is classified as -1. Finding a hyperplane is trivial, however to find the most optimal hyperplane it is best to maximize the margins between data points. We used the following equation to create a maximum margin hyperplane:

n – number of data points (images in our test bank)
w – line orthogonal to the hyperplane that corresponds to each data point
xi , yi – location of data points in 3D space
b – bias of hyperplane
λ - regularization constant (a constant that will prevent overfitting the data)

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|^2 \quad \textbf{(3)}$$

5. NOVELTY AND SIGNIFICANCE

The machine learning method learning by passing tests (LPT) is in itself a novel algorithm as its machine implementation of human-like learning by attempting and successfully passing tests of increasing difficulty was presented by Du and Xie [1] mere months ago. Additionally, the structure of our tester model is in itself novel. It is inspired by the algorithm described in LPT, but its implementation and architecture is very different and original. Instead of using the convolutional neural networks ResNet-18 and ResNet-50 [4] as experimented on by Du and Xie [1] to create increasingly challenging tests for the learner model we built our own automated unit test architecture driven by a Support Vector Machine which organizes and segments data of interest for test design. In LPT, the learner's architecture and weights are largely based on differentiable neural architecture search (DARTS) [1] whereas our learner model is a convolution neural network (VGG16) designed for image recognition [2]. Instead of using the CIFAR-10, CIFAR-100, and ImageNet datasets in the LPT experiment [1], we used an Alzheimer's dataset consisting MRI brain scans with four classes of Alzheimer's stages [3]. We feel the novelty of these differences to the LPT method and architecture presented enough of a challenge and reward of deeper understanding of LPT in the allocated schedule of this course.

6. RESULTS

While we enjoyed some success, our project did not produce as good of results as we had hoped for. While still converging to an accurate model, our algorithm was not appreciably better than standard methods and did not have any appreciable speed increase in training. The original model took approximately thirty minutes to train a model, and our model took about eight minutes to train with an average of four runs to have a well-performing model. In some cases the model generated was worse than standard methods. The original model produced an accuracy of 82%, and with traditional optimization techniques we were able to improve this to roughly 95%. Using our model we were able to improve the model's performance to 94.51% accuracy, matching traditional methods. There is further room for improvement on our design, and it may indeed function better with larger datasets or larger class diversity. This can only be shown through further experimentation with other datasets.

6.1. Experiments

All experiments start with the baseline model accuracy and confusion matrix depicted in Figure 8 below.

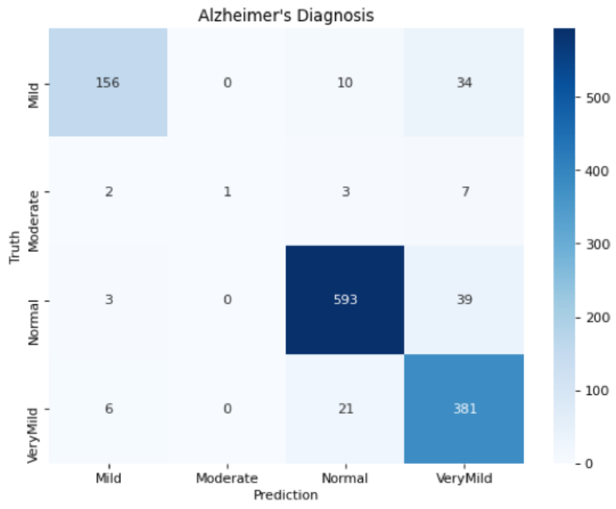

Fig. 8: Starting Model:
Train Accuracy: 88.17%
Validation Accuracy: 93.15%
Test Accuracy: 90.05%

6.1.1. Experiment 1

- 100 class1 and class2 train images chosen from SVM, divided into easy, medium and hard categories
- 100 class1 and class2 test images chosen from SVM, divided into easy, medium and hard categories
- Final results showed a 2.75 percent increase in test accuracy, shown in Figure 9

6.1.2. Experiment 2

- 500 class1 and class2 train images chosen from SVM, divided into easy, medium and hard categories
- 500 class1 and class2 test images chosen from SVM, divided into easy, medium and hard categories

- Final results showed a 3.63 percent increase in test accuracy, shown in Figure 10

6.1.3. *Experiment 3*

- 100 class1 and class2 train images chosen from SVM, divided into easy, medium and hard categories
- 100 class3 and class4 train images chosen from SVM, added to class1 and class2 easy, medium and hard categories
- 25 class1 and class2 test images chosen from SVM, divided into easy, medium and hard categories
- 25 class3 and class4 test images chosen from SVM, added to class1 and class2 easy, medium and hard categories
- Final results showed a 4.35 percent increase in test accuracy, shown in Figure 11

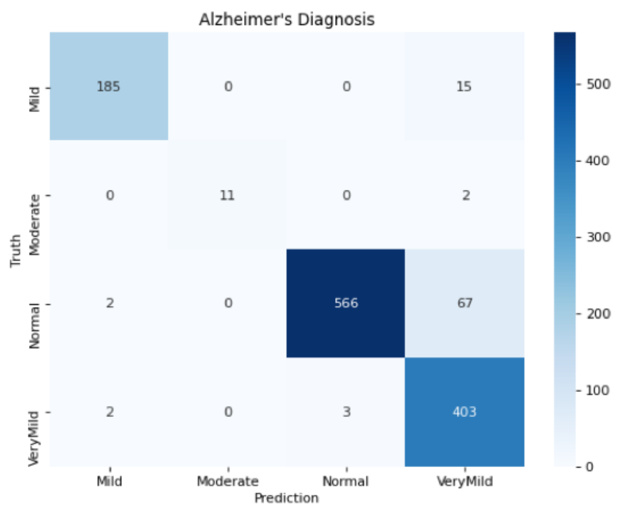

Fig. 9: Experiment 1 Results:
Train Accuracy: 98.98%
Validation Accuracy: 93.15%
Test Accuracy: 92.75%

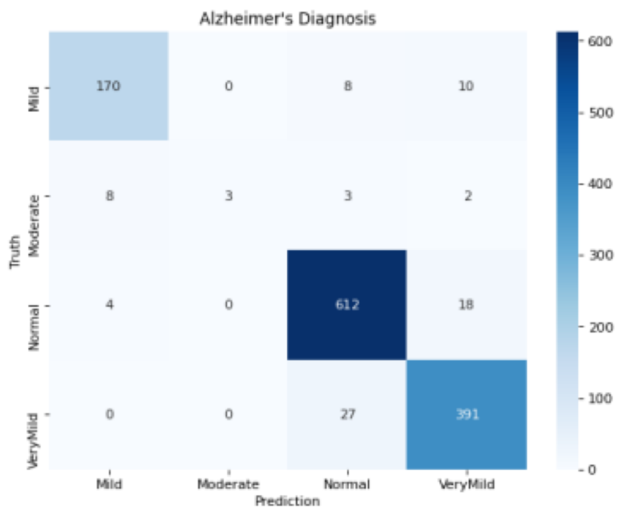

Fig. 10: Experiment 2 Results:
Train Accuracy: 99.68%
Validation Accuracy: 94.9%
Test Accuracy: 93.63%

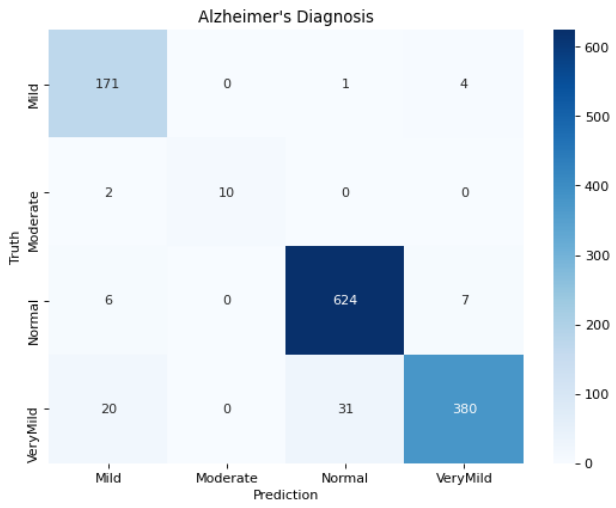

Fig. 11: Experiment 3 Results:
Train Accuracy: 99.54%
Validation Accuracy: 93.95%
Test Accuracy: 94.35%

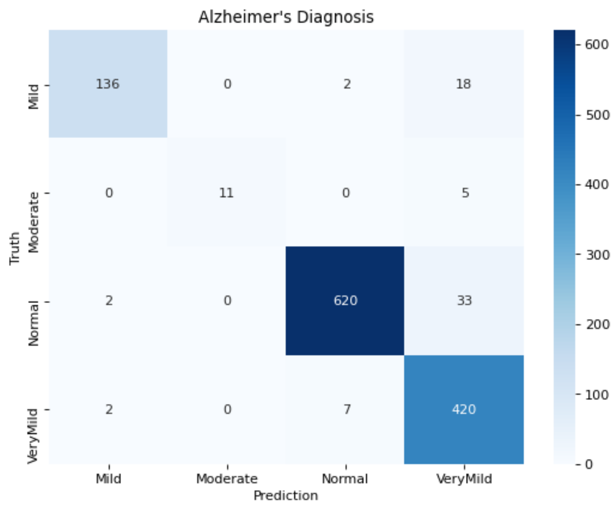

Fig. 12: Experiment 4 Results:
Train Accuracy: 99.41%
Validation Accuracy: 93.95%
Test Accuracy: 94.51%

6.1.4. Experiment 4

- 500 class1 and class2 train images chosen from SVM, divided into easy, medium and hard categories
- 500 class3 and class4 train images chosen from SVM, added to class1 and class2 easy, medium and hard categories
- 100 class1 and class2 test images chosen from SVM, divided into easy, medium and hard categories
- 100 class3 and class4 test images chosen from SVM, added to class1 and class2 easy, medium and hard categories
- Final results showed a 4.51 percent increase in test accuracy

6.2. Analysis of Results

We chose to use a dataset of roughly 6200 brain MRI scans labeled with 4 different stages of Alzheimer's disease, constituting four classes of data [3]. We anticipated the size and class diversity of this dataset would provide an acceptable variation in difficulty such that the tester model would successfully identify classification tasks in need of improvement and data samples appropriate for escalating levels of tests. However, with only four classes there were not many class similarities to choose from. This meant that the tests could only have six classification targets (4 choose 2) for improvement. We also feel that our approach did not have a very large set to pull targeted tests from, and would show much better results with larger datasets with more classes. In datasets with more classes the task of training models generally becomes more burdensome, and thus our targeted approach could be potentially more useful. Our model is also extremely RAM intensive as it entails repeatedly splitting the dataset to create subsets for training. This created problems in the environment we were using which had limited RAM allocation. This resulted in us saving data incrementally to external folders and running parts of the code in a piecemeal fashion. This made the project more unwieldy to coordinate between project partners, and caused the experiment to take extra time while running. This could be improved greatly by retaining only the original dataset and creating temporary subsets of the training data which could be cleared after use. We were unable to execute this efficiently in our project environment (Google Colaboratory) and would work to implement this in future iterations of this method. We are proud that our model performed well and increased the target accuracy, and will continue improving and exploring this novel approach.

7. PROJECT TIMELINE

1. Outline Learner/Tester Model
   - Jan 22 - 25
2. Develop Models/Tailor Dataset
   - Jan 25 - Feb 7
3. Evaluate and Revise Models
   - Feb 7 to Mar 10
4. Repeat Step 3 as Necessary
   - Ongoing
5. Create Final Report
   - Mar 10 - 17

8. REFERENCES

[1] Xuefeng Du and Pengtao Xie. *Learning by Passing Tests, with Application to Neural Architecture Search*. arXiv:2011.15102, Nov 2020.
`https://arxiv.org/abs/2011.15102`

[2] Karen Simonyan, Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv:1409.1556 (2015).
`https://arxiv.org/abs/1409.1556`

[3] Dubey, Sarvesh. *Alzheimer's Dataset (4 Class of Image)*. Kaggle (2019).
`https://www.kaggle.com/tourist55/alzheimers-dataset-4-class-of-images`

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385, Dec 2015.
`https://arxiv.org/abs/1512.03385`

[5] Jordan, Jeremy. *An Overview of Sematic Image Segmentation*. May 2018.
`https://www.jeremyjordan.me/semantic-segmentation/`

[6] *"Why Get Checked?"*. Alzheimer's Association, 2020.
`https://www.alz.org/alzheimers-dementia/diagnosis/why-get-checked`
`https://pubmed.ncbi.nlm.nih.gov/7164002/#:~:text=The\%20breathing\%20pattern\%20during\%20all,of\%2$the\%20level\%20during\%20wakefulness.`

[7] Visser, P.J., Scheltens, P., Verhey, F.R.J., Schmand, B., Launer, L.J., Jolles, J., Jonker, C. *Medial temporal lobe atrophy and memory dysfunction as predictors for dementia in subjects with mild cognitive impairment*. J. Neurol, pp. 246, 477–485, 1999

[8] Rathore, Habes, Iftikhar, Shacklett, Davatzikos. *A review on neuroimaging-based classification studies and associated feature extraction methods for Alzheimer's disease and its prodromal stages*. NeuroImage, 15, Jul, 2018.
`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5511557/`

[9] Csernansky, J.G., Wang, L., Swank, J., Miller, J.P., Gado, M., McKeel, D., et al. *Preclinical detection of AD: hippocampal shape and volume predict dementia onset in the elderly*. NeuroImage, 25, 783–792, 2005.

[10] Frisoni, Fox, Jack, Scheltens, Thompson. *The Clinical Use of Structural MRI in Alzheimer Disease*. Nature Reviews Neurology, Feb, 2010.
`https://www.nature.com/articles/nrneurol.2009.215`

[11] Plant, Teipel, Oswald, Bohm, Meindl, Mourao-Miranda, Bokde, Hampel, Ewers. *Automated detection of brain atrophy patterns based on MRI for the prediction of Alzheimer's disease*. NeuroImage, Mar, 2010.
`https://www.sciencedirect.com/science/article/pii/S1053811909012312?via\%3Dihub`

[12] S. Wang, H. Wang, Y. Shen and X. Wang. *Automatic Recognition of Mild Cognitive Impairment and Alzheimers Disease Using Ensemble based 3D Densely Connected Convolutional Networks*. 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, pp. 517-523, 2018.
`doi: 10.1109/ICMLA.2018.00083.`