

Alzheimer's Progression

an image classification project using MRI data

Mackenzie Austin, Emanuel Mejía, Ibrahim Shareef

Berkeley
UNIVERSITY OF CALIFORNIA

For our final project we decided to try an image classification problem with some MRI data on Alzheimer's progression.

Problem Statement | Objective

Non Demented

Very Mild Demented

Mild Demented

Moderate Demented

Non Demented

Non Demented

Non Demented

Non Demented

Non Demented

Very Mild Demented

Very Mild Demented

Very Mild Demented

Very Mild Demented

Very Mild Demented

Mild Demented

Mild Demented

Mild Demented

Mild Demented

Mild Demented

Moderate Demented

Moderate Demented

Moderate Demented

Moderate Demented

Moderate Demented

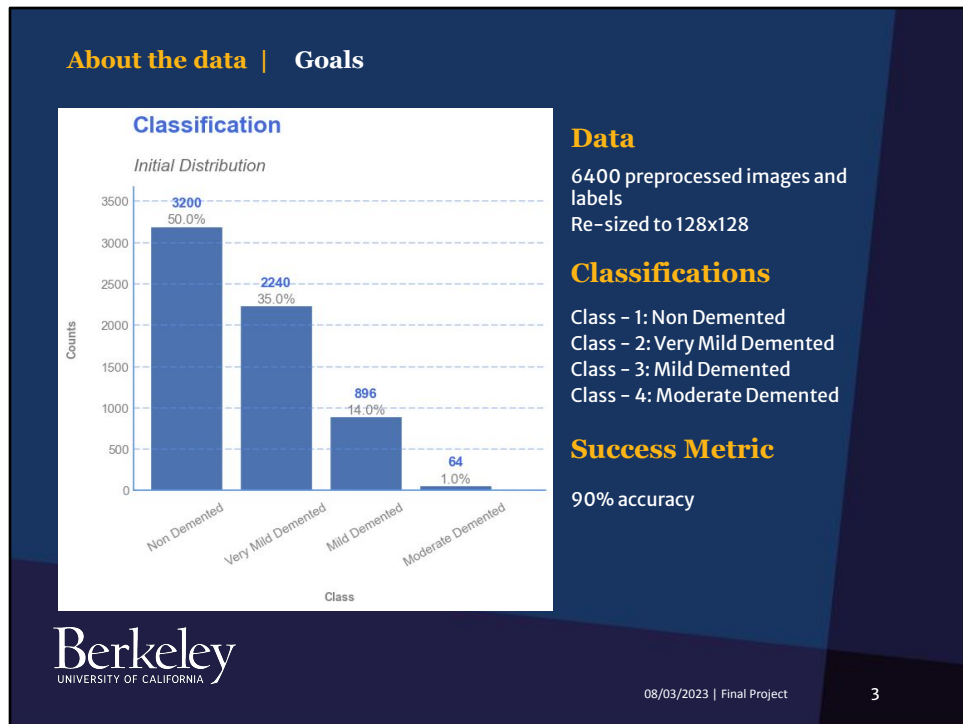
Berkeley

UNIVERSITY OF CALIFORNIA

08/03/2023 | Final Project

2

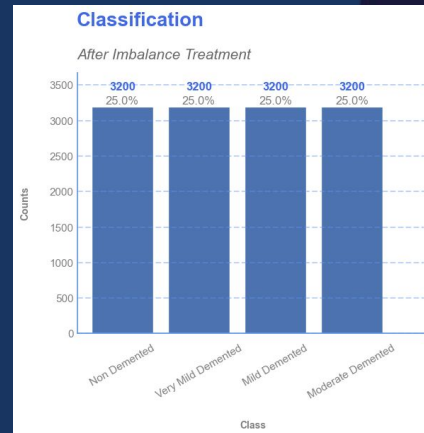
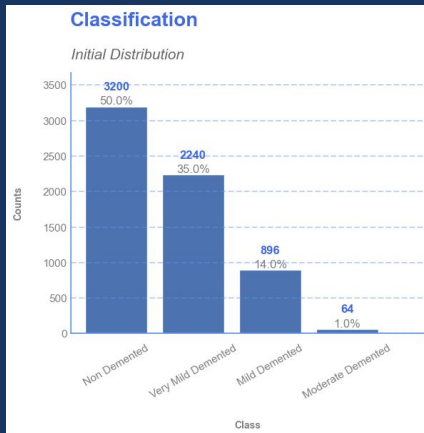
The process of getting diagnosed with Alzheimer's today is less than ideal, it involves different cognitive, medical and imaging tests that can consume many resources and valuable time. With the introduction of image classification software those costs can be reduced and patients can hopefully have earlier diagnoses allowing for better and more precise treatment. Additionally, if the accuracy of the classification can be increased, this software could potentially provide more accurate detection than even a doctor could.



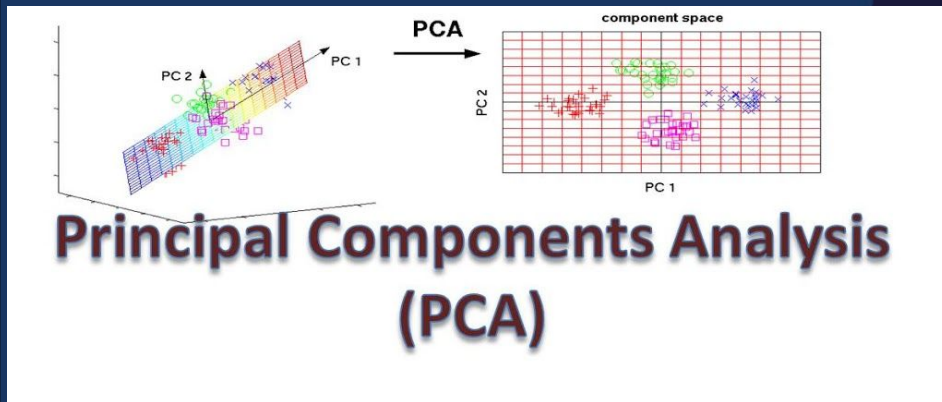
Taking a look into our dataset, we have a total of 6400 preprocessed labeled MRI images with 4 classifications, non-demented, very mild demented, mild demented, and moderate demented. These images have been resized to 128x128 pixels and standardized by dividing the pixels by 255. This chart on the left shows the distribution of our classifications, with most of the images being of non-demented brains..

We decided to be a little ambitious with this project and shoot for a success metric of 90% accuracy with our classification methods.

Data Engineering | Class Imbalance



Due to our significant class imbalance, we decided to use a module called SMOTE which is commonly used in other projects to correct class imbalances. This method uses data augmentation to generate synthetic data which have slight variations of the original data. This newly balanced data was then shuffled and split 80/20 for training and testing datasets.

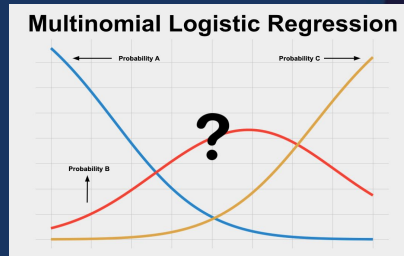
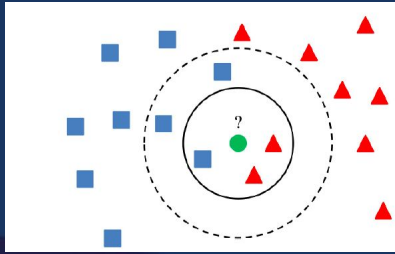


Another form of data engineering we performed was principal components analysis or PCA after experiencing some overfitting due to too many features in some of our models. PCA ensures we use only the most important and impactful features and maintain 85% of the variation of our data. Something we found rather interesting was that we reduced our features, or pixels in our case, from 49,152 down to 225, it was incredible to think that the majority of the variation of our data was contained in only 225 pixels. Throughout this project we will show both the results with and without PCA where appropriate.

Approach | Methodology

Classification Methods

- Multi-class Logistic Regression
- K-means clustering
- K nearest neighbor

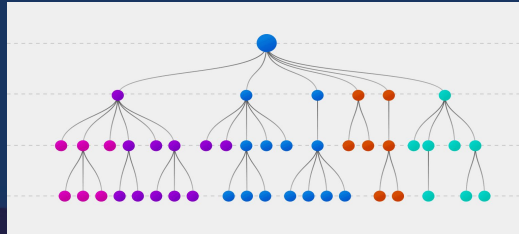
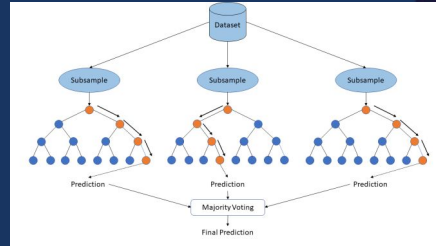


Next we're going to talk about the models we used for our classification problem. We employed a total of 7 machine learning models but for now let's discuss multiclass logistic regression, k-means clustering, and K nearest neighbors. Following common practice, we decided to use our multiclass logistic regression as our baseline and compare the other models against it. While multiclass logistic regression is sometimes used for image classification, our specific problem was a little too complex to be mapped to a linear relationship. K-means and K nearest neighbor are similar yet opposite models, both working through euclidean distance calculations, while K means starts with one cluster and splits into many, knn starts with every data point on its own and groups them by their neighbors, both having the capacity for overfitting.

Approach | Methodology

Classification Methods

- Decision Trees
- Random Forest



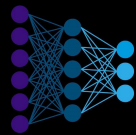
Moving onto our next two models, we have decision trees and random forest. Decision trees have a hierarchical tree-like structure consisting of decision nodes and are highly reliant on impactful and meaningful features of a dataset. Because our dataset includes images, our features are pixels, which can be difficult for decision trees to classify. Random forest performs similarly because it is a combination of many decision tree outputs. Both, when run long enough, have also the capacity of overfitting.

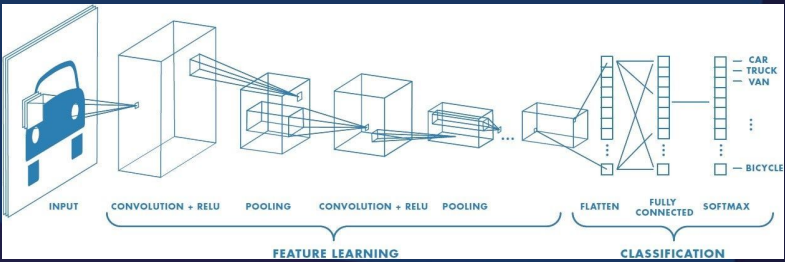
Approach | Methodology


Classification Methods

- Neural Networks
- Convolutional Neural Networks

Feedforward Neural Nets



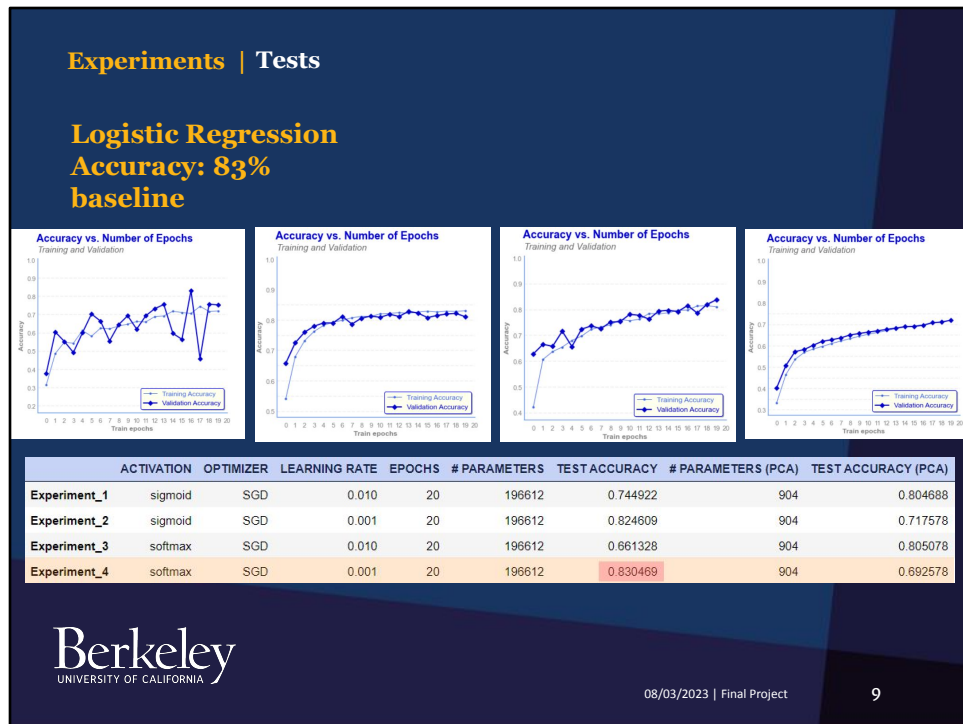




08/03/2023 | Final Project
8

Finally we have our last two models which performed the best, the neural nets. Both models aim to simulate the human brain and are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. The difference between the two here is that CNN has a specialized convolution layer which is meant to filter an image and produce a feature map, indicating the locations and strength of a detected feature in an input image.

Interestingly, however, the convolutional neural net performed worse than the non-convolutional in our case, but we'll cover that in a bit more detail later, for now let's dive into our experiments.



Because logistic regression is the simplest of our models, and following common practice, we will use the maximum accuracy of about 83% as baseline to compare against.

For each model we ran a series of experiments where various hyper parameters were adjusted. For this model we adjusted the activation function, and learning rate. As mentioned before we also performed every experiment with and without PCA. This table shows values of the hyper parameters and the test accuracy results.

We tried various combinations of sigmoid or softmax activation with SGD optimizers and learning rates of 0.01 or 0.001. And in the case of logistic regression PCA actually decreased accuracy occasionally. So our best performing logistic regression model was experiment 4: softmax activation, SGD optimizer, 0.001 learning rate, with no PCA. Resulting in an accuracy of 83%

K-means clustering

Accuracy: 84.6%
1.6% over baseline



Ok, now let's have some fun talking about how we tried to beat this baseline accuracy.

Our first model after the baseline is K-means clustering. We started by creating a different number of clusters and labeling each one with the class that's presented more frequently in it. Then each data point in the Testing Set would be part of the cluster it is closest to and labeled accordingly. In order to run as many experiments as needed to demonstrate its working we decided the number of clusters exponentially (from 1 cluster, on the top left of these confusion matrices, up to 512 clusters on the bottom right).

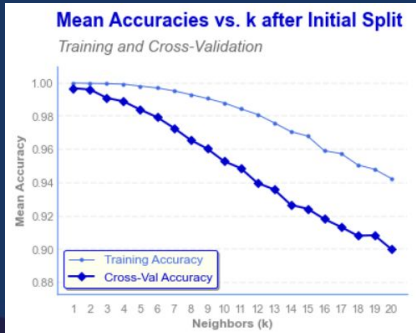
	# CLUSTERS	TRAIN ACCURACY	TEST ACCURACY
Experiment_1	1.0	0.252441	0.240234
Experiment_2	2.0	0.255664	0.244141
Experiment_3	4.0	0.287598	0.273828
Experiment_4	8.0	0.353613	0.354297
Experiment_5	16.0	0.523828	0.509375
Experiment_6	32.0	0.625781	0.618750
Experiment_7	64.0	0.652051	0.643750
Experiment_8	128.0	0.705762	0.688281
Experiment_9	256.0	0.799316	0.774219
Experiment_10	512.0	0.886035	0.846094

We can see in the next table that the higher the number of clusters, the higher the accuracy. And we could keep adding clusters up to the point of overfitting the data and not being able to generalize.

Experiments | Tests

KNN

Accuracy: 99.9% ~ 98.2%
16.9% ~ 15.2% over baseline



	NEIGHBORS (k)	CV ACCURACY	TRAIN ACCURACY	TEST ACCURACY
Experiment_1	1.0	0.99875	1.000000	0.999219
Experiment_2	2.0	0.998094	0.999005	0.998437
Experiment_3	3.0	0.991016	0.999609	0.994922
Experiment_4	4.0	0.988965	0.999316	0.992188
Experiment_5	5.0	0.983984	0.998145	0.981016
Experiment_6	6.0	0.979492	0.997070	0.988444
Experiment_7	7.0	0.973047	0.995313	0.983984
Experiment_8	8.0	0.965820	0.993066	0.982812
Experiment_9	9.0	0.960547	0.990723	0.977344
Experiment_10	10.0	0.953027	0.988086	0.977344
Experiment_11	11.0	0.948828	0.984473	0.969922
Experiment_12	12.0	0.939941	0.981152	0.965234
Experiment_13	13.0	0.936035	0.976074	0.960938
Experiment_14	14.0	0.926758	0.970801	0.955469
Experiment_15	15.0	0.924414	0.968164	0.950391
Experiment_16	16.0	0.918262	0.959277	0.945703
Experiment_17	17.0	0.913477	0.957817	0.939453
Experiment_18	18.0	0.908203	0.950781	0.935156
Experiment_19	19.0	0.908496	0.948145	0.928906
Experiment_20	20.0	0.900195	0.942676	0.925000

Moving onto our next algorithm, we have our classification k Nearest Neighbors model, which keeps dropping accuracy the more Neighbors we add to the model.

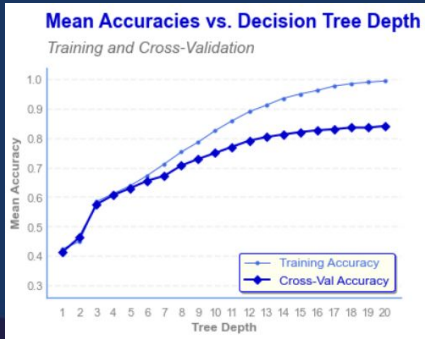
This was unexpected since our best result, even in the validation and testing sets, is when k equals 1. Which means it only takes our testing sample, looks for the closest point in the training dataset and labels it with the same class.

But, when we keep adding neighbors to the model, we find out that pictures inside it don't even have the same label, meaning that our training data points are not that distinct from one another, which can be confirmed on how the validation accuracy decreases at a much faster rate than training accuracy.

Meaning that as the previous model, we could be potentially overfitting our specific data and it would be highly risky when trying to generalize.

Experiments | Tests

Decision Tree
Accuracy: 86.7%
3.7% over baseline



	MAX DEPTH	CV ACCURACY	TRAIN ACCURACY	TEST ACCURACY
Experiment_1	1.0	0.415723	0.425684	0.413672
Experiment_2	2.0	0.465234	0.451367	0.452734
Experiment_3	3.0	0.576953	0.586035	0.579297
Experiment_4	4.0	0.608398	0.613281	0.614062
Experiment_5	5.0	0.631836	0.640137	0.635547
Experiment_6	6.0	0.657031	0.674902	0.668359
Experiment_7	7.0	0.673438	0.713770	0.691406
Experiment_8	8.0	0.708301	0.754395	0.709375
Experiment_9	9.0	0.730762	0.787891	0.724219
Experiment_10	10.0	0.750977	0.826855	0.758203
Experiment_11	11.0	0.771973	0.860254	0.787891
Experiment_12	12.0	0.792676	0.890625	0.811719
Experiment_13	13.0	0.804883	0.912793	0.816406
Experiment_14	14.0	0.813574	0.935937	0.833594
Experiment_15	15.0	0.821680	0.950977	0.841016
Experiment_16	16.0	0.828027	0.962793	0.839453
Experiment_17	17.0	0.831348	0.977637	0.856641
Experiment_18	18.0	0.837207	0.985938	0.856641
Experiment_19	19.0	0.836816	0.990918	0.866016
Experiment_20	20.0	0.842383	0.994922	0.867578

Our next model is another classification model: decision trees, where we kept running experiments adjusting the max depth of the tree.

We ran it up to a depth of 20 levels with a max test accuracy of about 87%, only a 4% increase compared to the baseline.

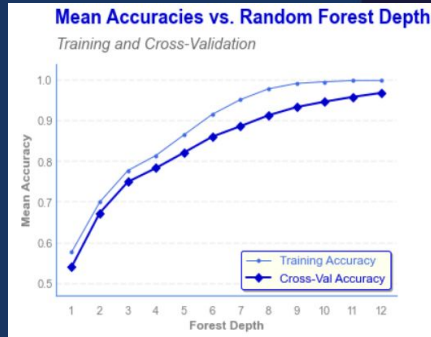
And we stopped there, because we can see that even when the accuracies are growing with each level we add to the tree, we're overfitting the data.

This is easily proven since the gap between training and validation accuracies keeps increasing in the graph on the left.

Experiments | Tests

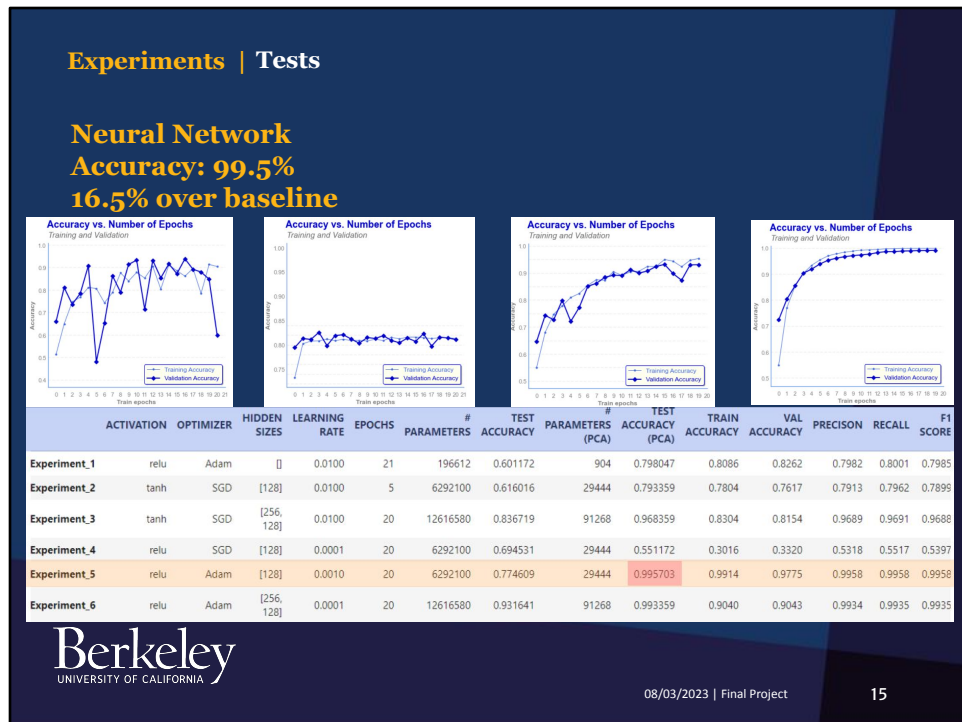
Random Forest
Accuracy: 97.2 %
14.2% over baseline

	MAX DEPTH	CV ACCURACY	TRAIN ACCURACY	TEST ACCURACY
Experiment_1	1.0	0.540332	0.576855	0.567187
Experiment_2	2.0	0.673438	0.700098	0.681641
Experiment_3	3.0	0.749902	0.777734	0.758203
Experiment_4	4.0	0.783984	0.815039	0.782031
Experiment_5	5.0	0.822168	0.865820	0.811719
Experiment_6	6.0	0.861035	0.916016	0.863672
Experiment_7	7.0	0.886816	0.952246	0.881641
Experiment_8	8.0	0.913672	0.978906	0.906250
Experiment_9	9.0	0.933984	0.992188	0.941016
Experiment_10	10.0	0.947266	0.996191	0.952344
Experiment_11	11.0	0.958984	0.999219	0.960156
Experiment_12	12.0	0.968750	0.999219	0.972266



We also ran a specific use of decision trees which is random forest. And as it's shown it has better results even using lower depths than with the trees. We only ran it up to 12 levels, and it got up to a 97% accuracy, which indicates that this could be potentially a good model.

We might keep adding levels to this forests' depth, although we need to be aware that this could potentially lead to the same overfitting problem we already discussed.



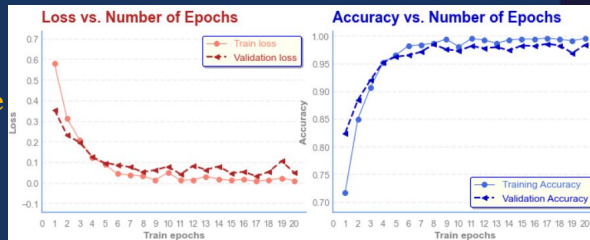
The next method we tried was a feed forward neural network, which ended up actually giving us the best results. So we ran many experiments after data augmentation, and before and after PCA was applied. This table shows a summary of the different experiments we ran and the various hyper parameters we were fine-tuning for each experiment. As shown in the table, we tried lots of different combinations of activation functions, optimizers, hidden layer sizes, learning rates, and number of epochs, to find the best model for our dataset.

We were eventually able to build a model that gave us a test accuracy of around 99.5%. This was almost 17% above our baseline with logistic regression

So we did worry a little bit because our accuracy was so high, that there was a possibility of overfitting because of the abnormally high accuracy in the training set as shown in the graph on the far right, but, because the validation accuracy is also just as high and we don't see a large difference between training and validation accuracy we ruled out this possibility because it still gave us a very high accuracy in our test set, which showed that it was still generalizing well on new data

Experiments | Tests

CNN
Accuracy: 98.3%
15.3% over baseline



	kernel size	strides	pool size	learning rate	optimizer	PARAMETERS	#	TEST ACCURACY	TRAIN ACCURACY	VAL ACCURACY	PRECISION	RECALL	F1 SCORE
Experiment_1	(5, 5)	(1, 1)	(2, 2)	0.001	Adam	647044.0		0.983594	0.9718	0.9785	0.9839	0.9837	0.9838
Experiment_2	(3, 3)	(1, 1)	(2, 2)	0.001	Adam	613764.0		0.977344	0.9485	0.9355	0.9775	0.9778	0.9776
Experiment_3	(5, 5)	(2, 2)	(2, 2)	0.001	Adam	122756.0		0.982813	0.9526	0.9531	0.9832	0.9832	0.9831
Experiment_4	(5, 5)	(1, 1)	(3, 3)	0.001	Adam	122756.0		0.972266	0.8760	0.8867	0.9727	0.9728	0.9727
Experiment_5	(5, 5)	(1, 1)	(2, 2)	0.010	Adam	647044.0		0.489453	0.4997	0.4961	0.3293	0.5000	0.3709
Experiment_6	(5, 5)	(1, 1)	(2, 2)	0.001	SGD	647044.0		0.782422	0.6095	0.6836	0.7823	0.7849	0.7827

And, finally we have our convolutional neural network, the model we expected to perform the best but it actually fell slightly behind our feed forward neural net. Again, we also ran lots of different experiments here and fine-tuned the different hyper-parameters. Parameters that were differing from neural nets that we adjusted for these experiments were kernel sizes, strides, and pool sizes. And here again we show a table that summarizes the results of each experiment.

Our best model with CNN gave us a test accuracy of around 98%. This was also true after data augmentation and PCA were applied.

Results | Evaluation

Accuracy $\frac{TP + TN}{\text{Whole Set}}$	Recall $\frac{TP}{TP + FN}$
Precision $\frac{TP}{TP + FP}$	F1-Score $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

Berkeley
UNIVERSITY OF CALIFORNIA

08/03/2023 | Final Project 17

Going over our evaluation metrics, while we were mainly using accuracy to measure the overall performance of our models, we also computed recall, precision, and F1-scores during our experiments. F1 scores are most helpful when the dataset is imbalanced like we initially had, but they were not significantly different from accuracy in our experiments because of the class balancing we performed with data augmentation.

For recall and precision, which we thought was also really important from a practical perspective, we found that it should perform particularly high against the more advanced classifications of dementia which were classified as mild to moderate. Here, we wanted to ensure our models had the lowest rate of false-negatives, or incorrect diagnoses when the patient was actually in an advanced stage of dementia.

Results | Evaluation

Neural Network

Mild Demented:

Precision: 98.7%

Recall: 99.7%

F1-score: 99.2%

Moderate Demented:

Precision: 100%

Recall: 100%

F1-score: 100%

CONFUSION MATRIX					TEST ACC: 96.84%
Non Demented	612	45	5	0	
Very Mild Demented	26	616	3	0	
Mild Demented	0	2	613	0	
Moderate Demented	0	0	0	638	
	Non Demented	Very Mild Demented	Mild Demented	Moderate Demented	
	Predicted Label				

Shown here are the precision, recall, and F1-scores for our best performing model in terms of accuracy which was our feed-forward neural network that I mentioned earlier. We specifically computed these scores for the mild and moderately demented classifications. As you can see from the matrix we scored 100% in all metrics for moderate demented, and above 98% for all metrics in the mild demented cases. Our model performed really well in these other metrics.

Time permitting, we would have liked to explore these evaluation metrics and confusion matrices for our other categories, but that is something we put for future work.

Constraints | Future Work

Limitations

- Limited in scope
 - Wealthy communities
 - Right handed
- Small dataset
- Not generalizable

Future Work

- More models
 - Transfer learning
 - LSTM
- Data collection
 - emphasis on fairness

Berkeley
UNIVERSITY OF CALIFORNIA

08/03/2023 | Final Project 19

There were many limitations for our project mostly centered around the data itself, compared to many other machine learning datasets, ours was relatively limited, in that 50% of the data was in one category, the non-demented case. While this is representative of the population, it does not facilitate good machine learning results. With more data, its possible no class imbalance would exists resulting in less biased results without resorting to SMOTE to fix imbalances.

Moreover, the data itself could be considered unfair in that it only contains data from wealthier areas, which is to be expected given the price of MRI scans but still yet not representative of the entire population that could be affected by this research. Additionally, the majority of the data is only for right handed people, which could have a significant effect on the image of the brain and subsequently the results of the research. For future work we want to focus on remedying the unfairness in the data by doing more data collection, attempting to perform MRI's on people of all communities and walks of life as well as left handed people.

Given the time we would also like to apply LSTM models, and further researching our evaluation metrics for the other two categories (non demented and very mild demented). We also want to consider the idea of transfer learning for future work, using pre-trained models as a framework and reference to compare and improve our models.

As we learn more and as the field of machine learning evolves, different models could be applied to our data to possibly produce better results.



Thank you

Data

<https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset/code>

Berkeley
UNIVERSITY OF CALIFORNIA

08/03/2023 | Final Project

20

Thank you everyone for listening!