CS 4341: Intro to AI
Project 2 Report

Adam Camilli
William Kelley
Skyler Kim
Feb 21 2018

Table of Contents

Project 2 Report

Neural Networks

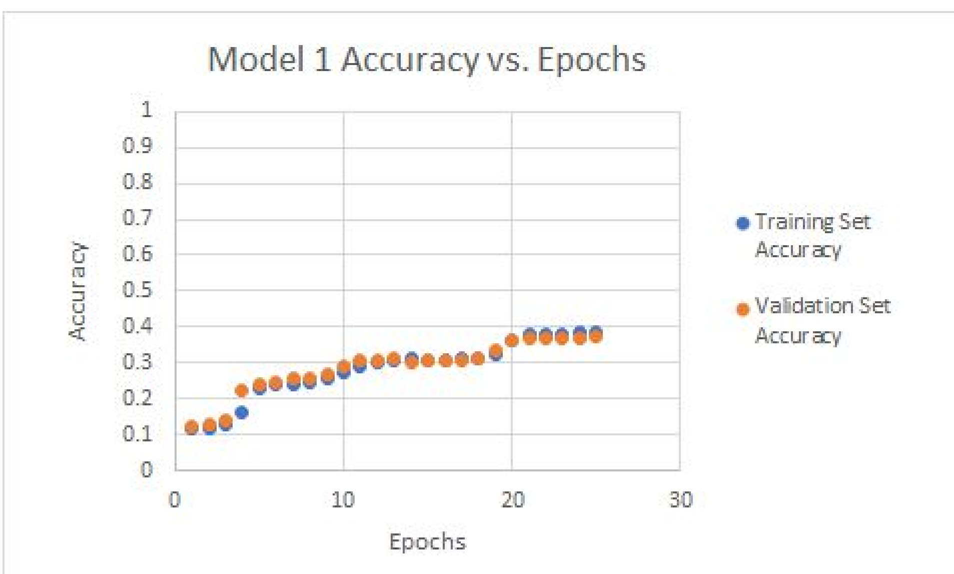
Model & Training Procedure Description

Our initial model consists of three layers, an input layer, a single hidden layer, and an output layer. The input layer is configured to take in a 28 by 28 matrix as the input and has ReLU as its activation procedure. The hidden layer then takes in the input and processes it with random weights before sending it to the output layer. Finally, the output layer uses the softmax activation procedure in order to present a fully parsable final output.

Our second model is similar to our first, but has an additional second hidden layer. Similarly, our third model is the same with a third hidden layer. All three hidden layers have neurons that use rectified linear units as their activation functions. We then compile the model and proceed to train it over 25 epochs using a batch size of 512. The training utilizes stratified data to ensure that the training set, validation set, and testing set have equal ratios of each type of number, while still retaining random sampling.

Graph

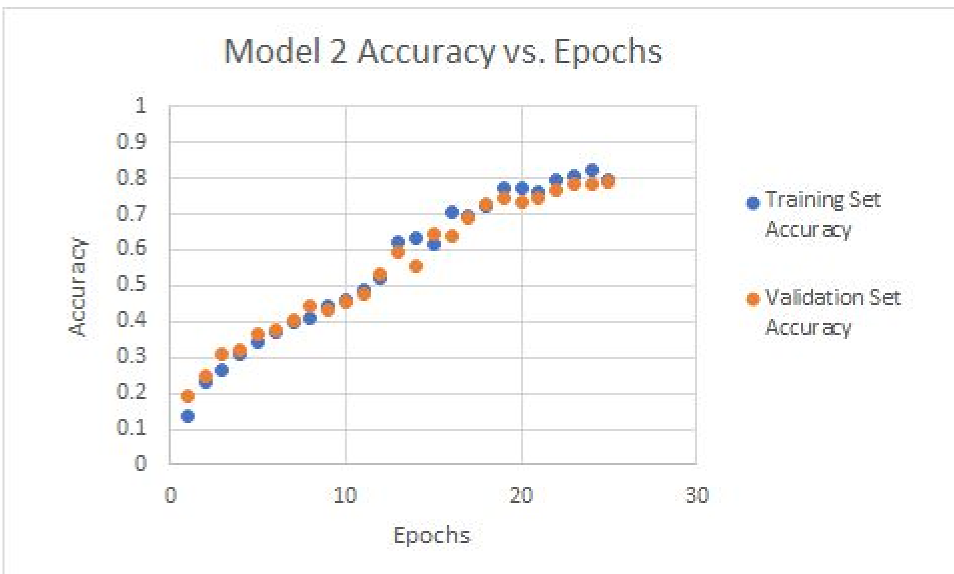
First Model Graph:



First Model Explanation:

From our results it is clear that the first model gradually increases in accuracy as additional epochs are added for training. Its growth, as seen in the graph, is quite similar to a step function. This implies that as our first model parses the information, it creates sharply delineated groups that get refined over time. If given enough epochs the model will eventually classify most but not all numbers correctly.

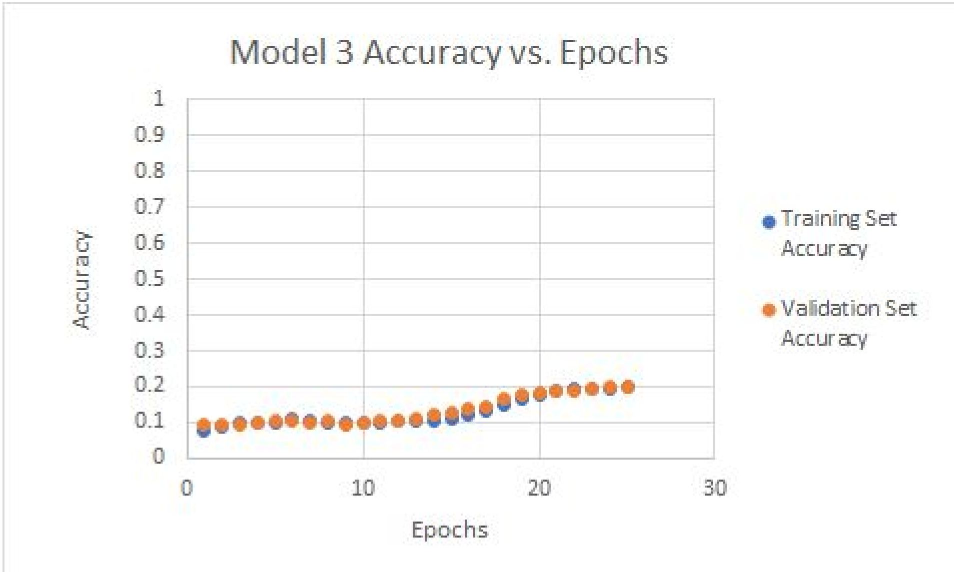
Second Model Graph:



Second Model Explanation:

From our results it is clear that the second model's rate of increase in accuracy is far more rapid with additional epochs for training. The second model seems to parse the data such that it creates a linear distribution which allows for rapid determination of what type a number is. In the short term, this allows the model to learn numbers very quickly but it also reaches its equilibrium not long after.

Third Model Graph:



Third Model Explanation:

From our results it is clear that the third model increases in accuracy much more slowly with additional epochs for training. The third model seems to create a sinusoidal distribution which requires additional time to learn the data. Due to learning with a high granularity the equilibrium point takes a long time to reach, but obtains a very high accuracy when it does.

Comparison:

Of our three models it is apparent that in the short term the second model is by far the best and the third model is by far the worst. With additional time and epochs to work with however, we found that the equilibrium points increase with the number of layers. That is, with enough epochs the third layer performs the best and the first layer performs the worst. This is shown in the following section. We believe this is because of how the models can fine-tune their parsing with additional layers.

Model Performance & Confusion Matrix

Performance For First model (25 Epochs):

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	161	0	0	0	0	0	2
1	0	175	0	5	0	0	0	0	0	2
2	0	20	1	127	0	0	1	4	0	6
3	0	3	1	147	1	0	1	6	0	9
4	0	5	0	128	0	0	0	0	0	31
5	0	2	0	131	1	0	0	4	0	4
6	0	5	0	157	0	0	2	0	0	2
7	0	1	1	3	0	0	0	142	0	25
8	1	3	1	137	0	0	0	1	0	7
9	0	1	0	11	0	0	1	12	0	136

First Model Explanation (25 Epochs):

The first model's results show that it was able to consistently guess images of 1's, 3's, 7's, and 9's. The accuracies of the model on these numbers were 96.15%, 87.50%, 92.56% and 84.47%, respectively. The remaining numbers were almost all incorrectly guessed as 3's. The total accuracy was 37.06%. It is possible that the model identified the rounded patterns of the number 3 and applied it to other numbers that can have rounded lines, such as 0's, 2's, 5's, 6's, and 8's.

Performance For First Model: (2000 Epochs):

	0	1	2	3	4	5	6	7	8	9
0	158	2	1	0	0	0	1	0	1	0
1	0	171	2	0	5	1	0	0	3	0
2	2	9	122	2	3	1	6	1	11	2
3	1	1	7	108	0	12	0	5	29	5
4	0	3	5	0	133	1	2	2	0	18
5	2	3	8	7	6	96	2	0	16	2
6	1	5	4	0	2	0	151	0	3	0
7	1	1	2	0	2	0	0	146	3	17
8	1	22	5	8	3	9	3	3	92	4
9	2	0	0	0	7	0	0	16	0	136

First Model Explanation (2000 Epochs):

Running the training data against 2000 epochs resulted in a significantly more accurate model. Predictions of 0's, 1's, and 6's had accuracies over 90%. Predictions for 4's, 7's, and 9's had accuracies between 80% and 90%. The remaining numbers had accuracies between 60% and 80%, with predictions for 8's being as low as 61.33%. The remaining images of 8's were usually incorrectly predicted as 1's, perhaps due to some 8's being thin enough that it disregarded their curvature. The total accuracy was 80.70%.

Performance For Second Model (25 Epochs):

	0	1	2	3	4	5	6	7	8	9
0	149	0	2	2	0	9	0	1	0	0
1	0	170	2	2	1	2	0	1	4	0
2	1	4	122	19	1	3	2	4	3	0
3	2	0	5	134	0	13	1	7	4	2
4	1	1	6	0	138	0	2	3	3	10
5	3	3	2	38	1	83	4	5	3	0
6	4	2	1	3	1	2	152	1	0	0
7	2	1	4	1	2	2	0	155	1	4
8	1	3	1	28	0	15	5	3	86	8
9	2	0	5	3	5	1	0	20	2	123

Second Model Explanation (25 Epochs):

The second model's results were fairly accurate. All predictions of images had accuracies above 75% except for images of 5's and 8's, where the accuracies were just above the 50% mark. Predictions for 0's, 1's, 6's, and 7's were above 90% accuracy. Images of 5's and 8's were usually predicted as 3's when predicted incorrectly. This was most likely due to similar curves being present in all three numbers. The total accuracy was 80.64%.

Performance For Second Model (2000 Epochs):

	0	1	2	3	4	5	6	7	8	9
0	153	0	1	2	0	4	0	2	1	0
1	0	168	0	1	1	1	2	1	6	2
2	3	3	132	5	2	4	6	2	2	0
3	1	3	8	131	0	15	1	2	6	1
4	1	0	1	0	138	2	4	4	1	13
5	7	0	1	9	2	107	6	0	8	2
6	2	0	6	0	8	4	146	0	0	0
7	3	5	2	5	3	1	0	145	1	7
8	2	2	3	10	1	6	7	2	115	2
9	1	1	0	2	10	2	0	11	1	133

Second Model Explanation (2000 Epochs):

Running the training data against 2000 epochs resulted in a significantly more accurate model. Predictions for images of 0's and 1's were over 90% accurate. Predictions for 3's, 5's and 7's were in the 70% to 80% accuracy range, while the remaining numbers were in the 70% to 80% accuracy range. The total accuracy was 84.08%.

Performance For Third Model (25 Epochs):

	0	1	2	3	4	5	6	7	8	9
0	0	101	0	0	0	0	22	12	0	28
1	0	160	0	0	0	0	16	5	0	1
2	0	121	0	0	0	0	16	9	0	13
3	0	152	0	0	0	0	10	3	0	3
4	0	5	0	0	0	0	155	0	0	4
5	0	90	0	0	0	0	34	7	0	11
6	0	8	0	0	0	0	153	2	0	3
7	0	18	0	0	0	0	149	4	0	1
8	0	38	0	0	0	0	96	4	0	12
9	0	3	0	0	0	0	154	2	0	2

Third Model Explanation (25 Epochs):

The third model was able to accurately predict 1's and 6's correctly, with accuracies above 85%. The remaining predictions were incorrectly listed as 1's or 6's. It is possible that the model started associating numbers with vertical lines with the number 1 and numbers with rounded lines with the number 6. The total accuracy was 19.61%.

Performance For Third Model (2000 Epochs):

	0	1	2	3	4	5	6	7	8	9
0	148	1	2	2	0	3	3	3	0	1
1	1	171	1	0	1	1	1	0	6	0
2	0	0	139	4	1	0	5	2	7	1
3	0	0	9	145	0	8	1	1	2	2
4	0	0	3	0	146	3	5	0	0	7
5	5	0	2	6	3	115	5	0	3	3
6	1	1	4	0	0	2	156	1	1	0
7	2	0	0	2	2	0	0	159	1	6
8	1	2	4	9	0	15	6	1	111	1
9	2	1	0	1	11	2	0	7	2	135

Third Model Explanation (2000 Epochs):

Running the training data against 2000 epochs resulted in a significantly more accurate model. Predictions for images of 0’s, 1’s, 6’s, and 7’s had prediction accuracies of over 90%. Predictions for images of 2’s, 3’s, 4’s, 5’s, and 9’s were in the 80% to 90% accuracy range. The only predictions with less than 80% accuracy were those for 8’s, which had a prediction accuracy of 74%. The total accuracy was 87.58%.

Comparison Between Models:

The second model with two rectified linear unit layers at 25 epochs had considerably higher accuracies than the first model with one rectified linear unit layer and the third model with three rectified linear unit layers. However, at 2000 epochs, the third model would be considered the best, with more numbers having a prediction accuracy over 90% and being one of the models with the highest amount of accurate predictions in the 80% to 90% range. Comparing the results for the second model and the third model, it is possible that the number of hidden rectified linear unit layers in the artificial neural network had an effect on how well the artificial neural network could learn for a given number of epochs. With the smaller number of epochs, the second model was more accurate than the third model. However, with a large amount of epochs, the third model was slightly more accurate than the second model. This might mean that more hidden layers requires more epochs for the model to fully learn the training set. However, giving the model more time to learn the training set would result in a higher accuracy than if less layers were used. So it is better to have less layers in the artificial neural network if only a small number of epochs can be run, but it is better to have more layers if there is more time and data allowed for more epochs.

Neural Network:	Accuracy:
Model One 25 Epochs	0.3706207744
Model One 2000 Epochs	0.8070067609
Model Two 25 Epochs	0.8063921328
Model Two 2000 Epochs	0.8408113092
Model Three 25 Epochs	0.1960663798
Model Three 2000 Epochs	0.8758451137

Visualization

Misclassified Image One:

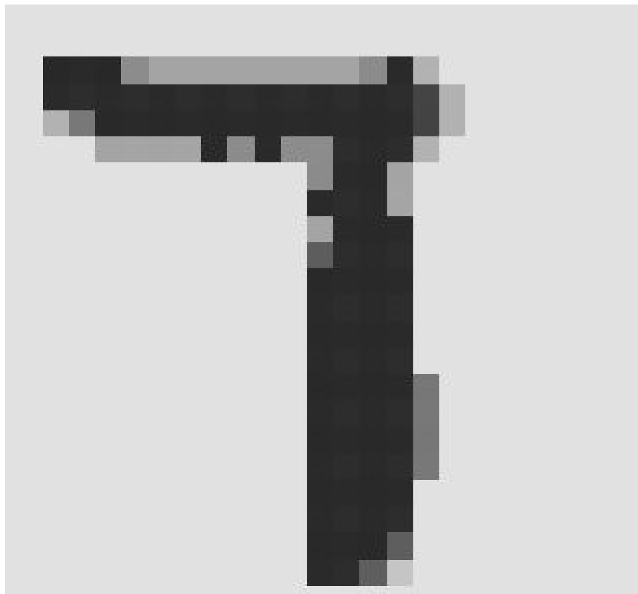


Image One Explanation:

Our first model misclassified this 7 as a 9. This could be because our first model made the delineation that any drawing with a long vertical line and a curved horizontal line at the top was a 9, which is a reasonable assumption to make.

Misclassified Image Two:

Image Two Explanation:

Our second model misclassified this 4 as a 9. This could be because the 4 is slanted and the portion extended from the top left is slightly curved. Additionally, if the two top protrusions were connected the number would look more like a 9 than a 4.

Misclassified Image Three:

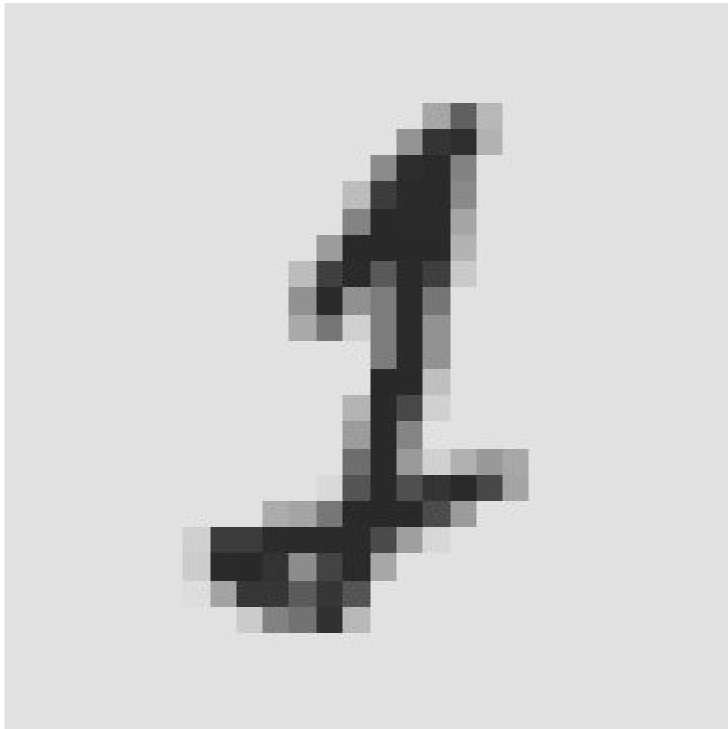


Image Three Explanation:

Our third model misclassified this 2 as a 1. This is likely because the number looks more like a slanted 1 than a 2. A number of humans who witnessed this image thought it was a 1. Even though the model misclassified the number, we believe it is a valid judgement.

Decision Trees

Description of Experiments

The initial model was a decision tree that treated each pixel in the training data set as a separate feature. The input for training the decision tree is a series of 28 by 28 pixel images that are squashed down to be a series of one dimensional arrays of length 784. The same method for obtaining a random stratified sample was used from the previous section. There was no specified depth limit to the tree, so the tree trained itself until it reached 100% accuracy for the training set. This model was then tested against the validation data set.

The subsequent models tested the effect of limiting the depth of the decision tree during the training of the model. The tree was limited to 3, 4, and 5 layers for the next three models per the suggestion of the project description, in comparison to the original model, which generated a tree of depth 18.

Feature Extraction & Explanation:

Our final decision tree makes use of the following four features where shaded pixel refers to a pixel with a grayscale value greater than half the maximum possible value:

- Longest vertical string of shaded pixels
- Longest horizontal string of shaded pixels
- Shortest vertical string of shaded pixels
- Shortest horizontal string of pixels

The feature longest vertical string of shaded pixels is meant to help the tree differentiate between numbers that are supposed to have straight edges and numbers that have rounded edges but are sometimes written as near vertical lines. This is usually the case with 3's and 5's.

The feature longest horizontal string of shaded pixels is meant to help the tree correctly label numbers that are supposed to have a long horizontal line. These numbers, such as 2's and some 1's, are usually classified as numbers with rounded edges on the bottom, like 6's and 8's, because the long horizontal line in the number is identified as a curve.

The features shortest horizontal string of shaded pixels and shortest vertical string of shaded pixels is meant to help the tree differentiate between numbers that are mostly curves. Numbers that consist mostly of curves, like 8's and 3's, do not have a lot of long strings of shaded pixels. At the same time, these are different from numbers that consist of curves and straight lines, like the numbers 2 and 5.

Model Performance & Confusion Matrix

First Model (No Depth Limit):

	0	1	2	3	4	5	6	7	8	9
0	84	0	3	3	0	1	1	3	2	1
1	0	95	3	3	1	1	3	2	1	1
2	1	3	73	5	2	0	6	1	3	2
3	1	0	1	78	2	10	0	1	2	5
4	0	1	0	1	79	3	0	5	2	7
5	2	0	1	1	2	66	1	4	7	2
6	2	2	2	1	1	2	83	1	6	0
7	0	4	0	2	2	1	1	90	1	2
8	3	4	6	8	1	10	1	1	50	6
9	1	0	3	5	3	5	0	7	1	72

Explanation of First Model

Running the data without a depth limit reported fairly accurate results. The total accuracy of the model on the valid data was 78.73%. Predictions for 0,1,4,6,7 were above 80% accuracy but under 90%. Predictions for for 2,3,5,9 were above 70% accuracy but below 80%. The remaining number was under 60% accuracy. The high accuracy here is most likely due to the fact that the tree was exhaustive in learning the data.

Second Model (Depth Limit = 3):

	0	1	2	3	4	5	6	7	8	9
0	82	0	0	12	0	0	0	4	0	0
1	0	105	0	2	0	0	0	3	0	0
2	5	21	18	38	0	0	1	13	0	0
3	13	9	3	64	0	0	0	11	0	0
4	2	6	1	9	0	0	0	80	0	0
5	12	17	0	25	0	0	3	29	0	0
6	3	14	2	13	0	0	60	8	0	0
7	4	9	0	1	0	0	0	89	0	0
8	0	35	0	50	0	0	0	5	0	0
9	3	6	0	6	0	0	0	82	0	0

Explanation of Second Model:

Running the data with a depth limit of 3 gave inaccurate results in comparison to the baseline model. The overall accuracy of the model was 42.74%. Predictions for 1 were the only predictions with an accuracy of over 90%. Predictions for 0 and 7 were between 80% and 90%. Predictions for 3 and 6 were over 60%, but under 80%. Predictions for 2 were under 18.75%. The remaining numbers had 0% prediction accuracies. It makes sense that this model is not nearly as accurate as the baseline model because the max depth of this model is lower than the depth limit of the baseline model. This results in less granularity of type groupings.

Third Model (Depth Limit = 4):

	0	1	2	3	4	5	6	7	8	9
0	80	1	2	1	2	0	0	11	1	0
1	0	95	8	1	1	0	1	1	3	0
2	3	7	64	5	7	0	0	3	7	0
3	8	6	12	63	3	0	0	3	5	0
4	0	2	4	12	66	0	0	8	6	0
5	7	3	18	37	8	0	1	7	5	0
6	3	1	10	2	10	0	57	5	12	0
7	1	1	4	1	7	0	0	85	4	0
8	0	7	22	5	6	0	0	3	47	0
9	0	4	9	26	39	0	0	15	4	0

Explanation of Third Model:

Running the training data with a depth limit of 4 resulted in a total accuracy of 56.95%. Predictions for 0, 1, and 8's had an accuracy between 80% and 90%. Predictions for 2, 3, 4, 6, and 8 were between 50% to 70% accurate. The predictions for 5 and 9 were 0% accurate. The accuracy of the model is 14.21% more accurate than the previous model, but the low max depth still limits its accuracy.

Fourth Model (Depth Limit = 5):

	0	1	2	3	4	5	6	7	8	9
0	77	0	0	1	2	10	0	6	0	2
1	0	100	6	1	2	0	0	1	0	0
2	1	5	58	1	5	5	7	1	9	4
3	0	3	5	59	0	20	1	3	5	4
4	3	0	2	3	75	2	1	2	1	9
5	2	0	1	2	1	71	2	1	2	4
6	7	1	3	1	5	9	63	1	8	2
7	0	0	0	9	5	2	0	71	2	14
8	2	1	7	1	5	10	3	0	51	10
9	0	0	1	1	6	11	0	2	0	76

Explanation of Fourth Model:

The accuracy of the fourth model was 71.68% accurate with regard to the validation data set. Predictions for 1's were over 90% accuracy. Predictions for 5's were between 80% and 90% accuracy. Predictions for 0, 4, and 9's were 70% and 80% accurate. Predictions for 2, 6, and 7 were between 60% and 70% accurate. Predictions for 3 and 8 were between 50% and 60%.

Comparison Between Depth Limiting Models:

It seems obvious that increasing the max depth limit of the decision tree would increase the accuracy of the model. Adding another layer to the decision tree would increase the amount of groups the original data could be broken up into. What is more interesting is how much the accuracy of the model changed as the depth limit was increased. The depth limiting sample of 5 layers had an accuracy of 71.68%. This is a 7.05% difference from the baseline model, which had 18 layers. So the rate of increase of accuracy with regard to how deep the decision tree is rapid at first, and becomes slower as the tree reaches its exhaustive size. At the same time, having an exhaustive tree is not necessary to have an accurate model. This experiment proves that having a tree with only one third of the depth of the exhaustive tree is sufficient enough to get an accurate model.

Fifth Model (Custom Features):

	0	1	2	3	4	5	6	7	8	9
0	67	0	2	0	1	2	4	9	5	8
1	0	96	1	2	0	0	1	1	4	5
2	4	1	52	3	2	0	5	3	14	12
3	5	1	5	58	4	1	2	6	7	11
4	2	1	1	1	66	2	6	3	3	13
5	3	0	1	20	2	26	4	11	4	15
6	2	0	7	0	7	0	71	1	1	11
7	0	0	1	0	6	0	0	80	0	16
8	1	2	1	6	1	5	3	1	62	8
9	0	0	1	2	10	0	1	5	1	77

Explanation of Fifth Model:

The overall accuracy of the fifth model was 66.9734%. Predictions for 1's were were the highest with over 80% accuracy but below 90% accuracy. Predictions for 6, 7, and 9's had an accuracy between 70% and 80%. Predictions for 0, 4, and 8's were between 60% and 70%. Predictions for 2 and 3's were between 50% and 60%. Predictions for 5's were the lowest, with an accuracy between 30% and 40%

Comparison Between Fifth Model and Fourth Model:

The model that looked at our extra features had a lower accuracy than the model with an equivalent depth limit that only trained on raw pixel data. It is possible that looking at only straight strings of pixels caused the accuracy of numbers that primarily consist of loops to decrease. This can be seen the in 50.33% accuracy decrease between predictions of 5's for the fourth model and the predictions of 5's in the fifth model. However, the increase in numbers that have straight lines, such as 6's and 7's, did see an increase of about 8% of accuracy.

Comparison Between Neural Networks and Decision Trees:

Overall decision trees tended to be less accurate than neural networks. This is due to decision trees being unsuited to handling the many edge cases present in the provided data set.

Model:	Accuracy:
First Model (Baseline)	0.7873210634
Second Model (Depth Limit = 3)	0.4274028630
Third Model (Depth Limit = 4)	0.5695296524
Fourth Model (Depth Limit = 5)	0.7167689162
Fifth Model (Custom Features)	0.6697341513

Neural Network:	Accuracy:
Model One 25 Epochs	0.3706207744
Model One 2000 Epochs	0.8070067609
Model Two 25 Epochs	0.8063921328
Model Two 2000 Epochs	0.8408113092
Model Three 25 Epochs	0.1960663798
Model Three 2000 Epochs	0.8758451137

KNN

Description of Experiments

For our experiments we created a system that made use of a k-nearest neighbor classification and altered the value of k to find different results. We used the values 1, 3, 6, and 12. For our findings we obtained the following accuracy values.

k-Value:	Accuracy:
1	0.9293177628
3	0.9360786724
6	0.9059618931
12	0.8832206515

Here we can see a trend emerges where as k increases, the accuracy decreases. This makes sense, as increasing k means that our algorithm will accept inputs that are further away from the exact value which invites more possibilities for error.

Model Performance & Confusion Matrix

Performance For k = 1:

	0	1	2	3	4	5	6	7	8	9
0	158	1	0	2	0	0	1	1	0	0
1	0	177	1	0	1	0	0	2	0	1
2	0	1	151	2	0	0	1	2	2	0
3	1	1	2	151	0	6	1	2	1	3
4	0	2	1	0	146	1	2	0	0	12
5	2	1	0	2	0	130	5	0	2	0
6	0	1	0	0	0	0	165	0	0	0
7	0	5	0	0	2	0	0	158	0	7
8	0	2	2	4	1	3	1	2	131	4
9	0	1	2	2	5	0	0	6	0	145

Performance For k = 3:

	0	1	2	3	4	5	6	7	8	9
0	163	0	0	0	0	0	0	0	0	0
1	1	180	0	0	0	0	0	1	0	0
2	6	7	138	1	0	0	1	2	4	0
3	2	0	1	156	0	6	1	2	0	0
4	0	3	0	0	150	0	1	0	0	10
5	2	1	1	1	0	134	1	0	0	2
6	4	2	0	1	0	0	159	0	0	0
7	3	3	1	0	2	0	0	157	0	6
8	6	5	0	1	0	1	0	0	134	3
9	3	2	0	1	1	0	0	2	0	152

Performance For k = 6:

	0	1	2	3	4	5	6	7	8	9
0	162	1	0	0	0	0	0	0	0	0
1	2	178	0	0	0	0	0	1	0	1
2	12	6	139	0	0	0	0	1	1	0
3	11	3	0	150	0	2	0	0	1	1
4	11	1	0	0	144	0	1	0	0	7
5	7	1	0	1	0	132	0	0	0	1
6	2	0	0	0	0	0	164	0	0	0
7	11	5	0	0	1	0	0	149	0	6
8	31	3	0	2	0	3	0	0	110	1
9	11	1	0	0	0	0	0	3	0	146

Performance For k = 12:

	0	1	2	3	4	5	6	7	8	9
0	162	0	0	0	0	0	1	0	0	0
1	1	179	1	0	0	0	0	1	0	0
2	16	10	125	0	0	0	0	5	2	1
3	11	6	0	148	0	2	0	1	0	0
4	12	5	0	0	138	0	1	0	0	8
5	20	0	0	1	0	119	2	0	0	0
6	6	2	0	0	0	1	157	0	0	0
7	12	3	1	0	2	0	0	151	0	3
8	27	4	0	0	0	4	2	0	111	2
9	11	0	0	0	1	0	0	2	0	147

Comparison Between k-values:

The k-values, in the context of the k-nearest neighbors algorithm, represent how much variation from a known value that our algorithm pair with that value. That means with a larger k-value the algorithm will accept more variation which means while more edge cases will be accepted, there is also a higher chance to accept a wrong value. Given the huge visual variation in each of the images, there are such a variety of edge cases that it causes the main value for every number to be within several deviations from the edge cases of the other numbers. As a result, as k increases, the amount of miscategorized edge cases also increases.

Comparison Between Algorithms:

Of the three types of algorithms we implemented, we found that k-nearest neighbors was the most accurate. Properly trained neural networks are slightly more accurate than the best decision trees. This is largely due to the various suitabilities of the algorithms to this particular problem. The problem the algorithms are tackling is a classification problem with many edge cases. As a result k-nearest neighbors, which is concerned with grouping and handles edge cases well, does the best. Neural networks come in second due to their flexibility when it comes to a variety of problem types. Decision trees come in last due to their inability to handle edge cases well.

Neural Network:	Accuracy:
Model One 25 Epochs	0.3706207744
Model One 2000 Epochs	0.8070067609
Model Two 25 Epochs	0.8063921328
Model Two 2000 Epochs	0.8408113092
Model Three 25 Epochs	0.1960663798
Model Three 2000 Epochs	0.8758451137

Model:	Accuracy:
First Model (Baseline)	0.7873210634
Second Model (Depth Limit = 3)	0.4274028630
Third Model (Depth Limit = 4)	0.5695296524
Fourth Model (Depth Limit = 5)	0.7167689162
Fifth Model (Custom Features)	0.6697341513

k-Value:	Accuracy:
1	0.9293177628
3	0.9360786724
6	0.9059618931
12	0.8832206515

Visualization

Misclassified Image One:

Image One Explanation:

Our algorithm using the value k = 1 misclassified this 3 as a 2. This misclassification could have occured due to the 3 looking like a 2 if you remove the top protrusion.

Misclassified Image Two:

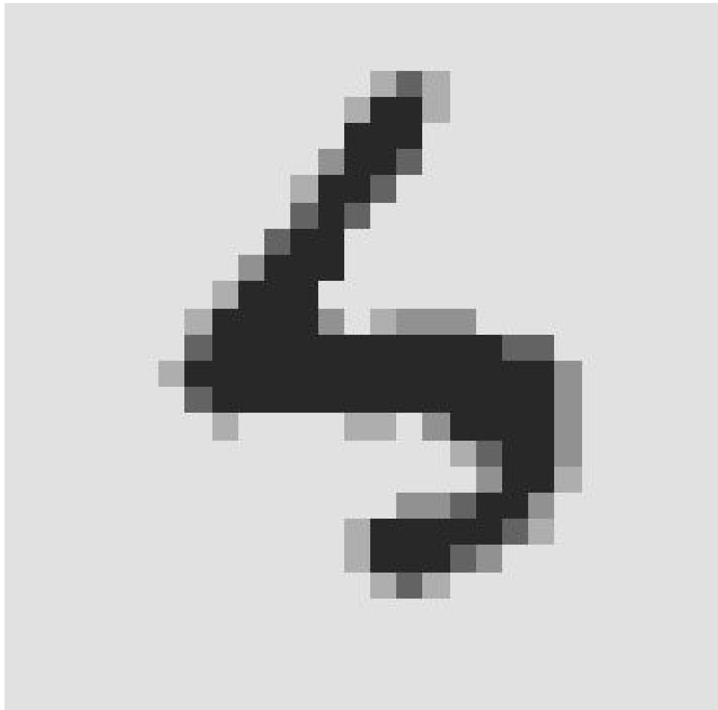


Image Two Explanation:

Our algorithm using the value k = 3 misclassified this 5 as a 3. This likely due to the orientation of the 5, since the algorithm views the image vertically, where it does resemble an incomplete 3.

Misclassified Image Three:

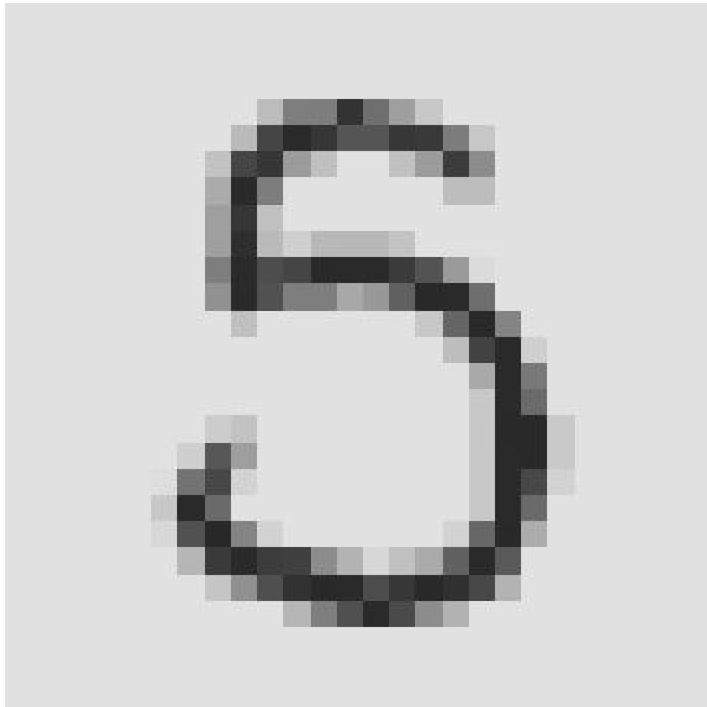


Image Three Explanation:

Our algorithm using the value k = 6 misclassified this 5 as a 6. Most likely it was fooled by the continuous curve in the image, which fits the profile of how a 6 is usually written and not the profile of a 5, which is usually drawn with a horizon top.