Kyle Savell
Henry Wheeler-Mackta
Richard Valente
Intro to AI
Project 2

Creation and Testing of Machine Learning Algorithms

Introduction:

Using a set of 6500 images and labels, 3 different machine learning algorithms were used to teach the program handwritten numbers. The algorithms implemented were an Artificial Neural Network using Keras, a Decision Tree using Scikit-learn, and a K-Nearest Neighbors algorithm also using Scikit-learn. The set of 6500 images were split into a training set (60% of the data), a validation set (15% of the data) and a test set (25% of the data). Data was NOT split into stratified samples, which could be a caveat of the experiment. Below are the results based on the three algorithms and variations thereof.

Artificial Neural Network:

Model & Training:

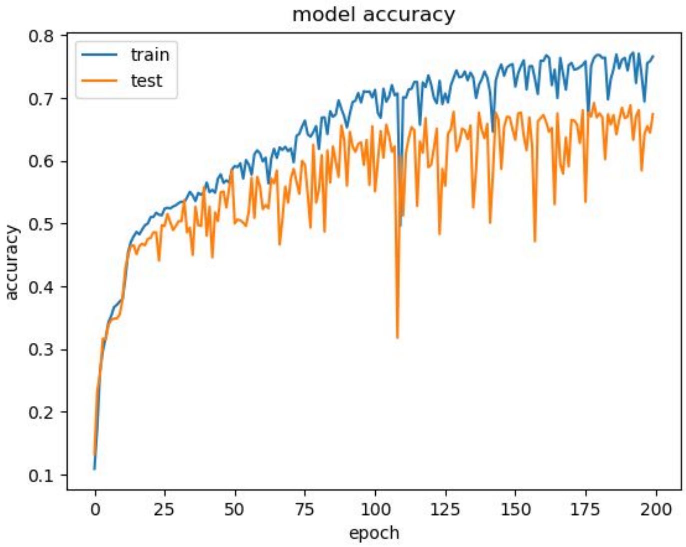
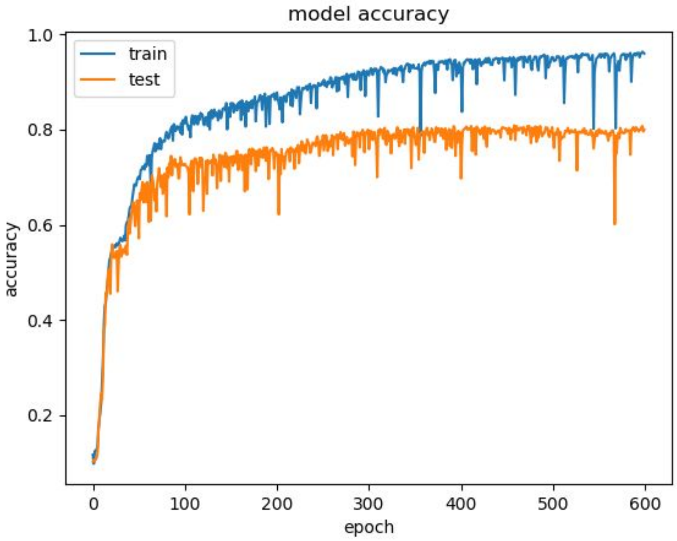
The neural network was created using the *keras* and *numpy* packages. The best accuracy achieved as 81.41%. Three variables were investigated to find a more optimal setup for the neural network, epoch, number of hidden layers and batch size. The optimal setup was 1500 epochs with 3 hidden layers with a batch size of 256.

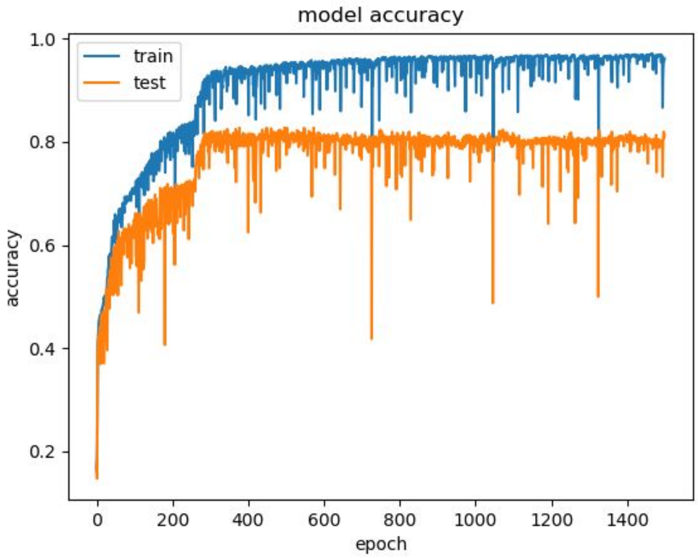
During experimentation and training a few things were observed even with the optimal setup, one of them being that the final accuracy was highly dependant on the initial accuracy of the randomly initialized network. We experimented with smaller batch sizes but training times were either extremely slow or stayed at sub-optimal loss minima. A batch size of 256 allowed for the function to “jump” out of local minima towards the global minima.

The following graphs display various tested values for epoch, batch size and hidden layer numbers.

Epoch:

Batch Size: 256, Hidden Layers = 3

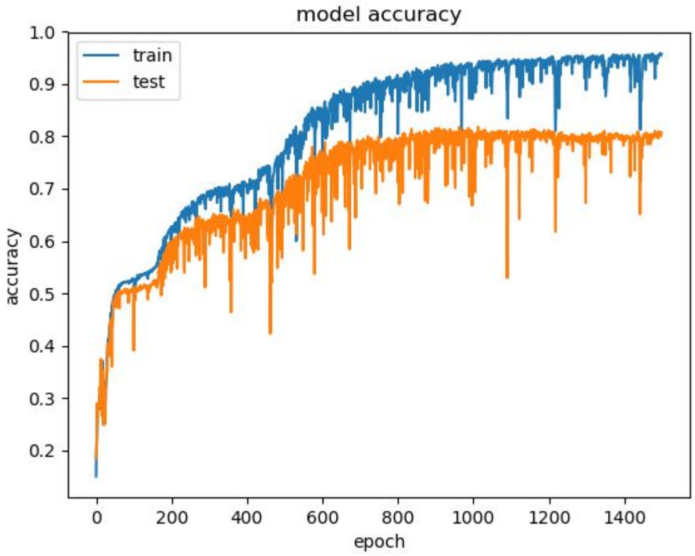
Epoch = 200: Max Validation Accuracy: 67.44%	
Epoch = 600: Max Validation Accuracy: 80.00%	

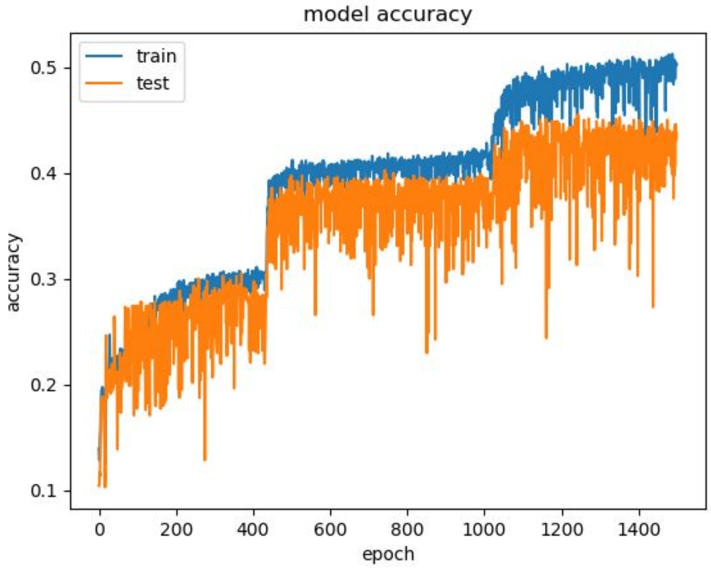
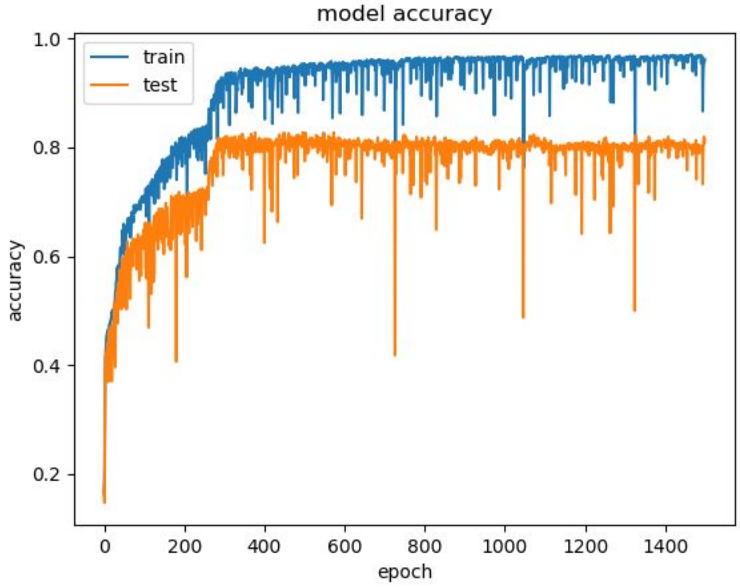
Epoch = 1500: Max Validation Accuracy: 81.41%	

Summary for Epoch:

The trend for epoch was increased accuracy as more epochs was ran., there were only slight gains in accuracy from 600-1500 of 1.41%.

Hidden Layers:
Setup: Batch Size = 256, Epoch = 1500

One Hidden Layer: Max Validation Accuracy: 79.96%	

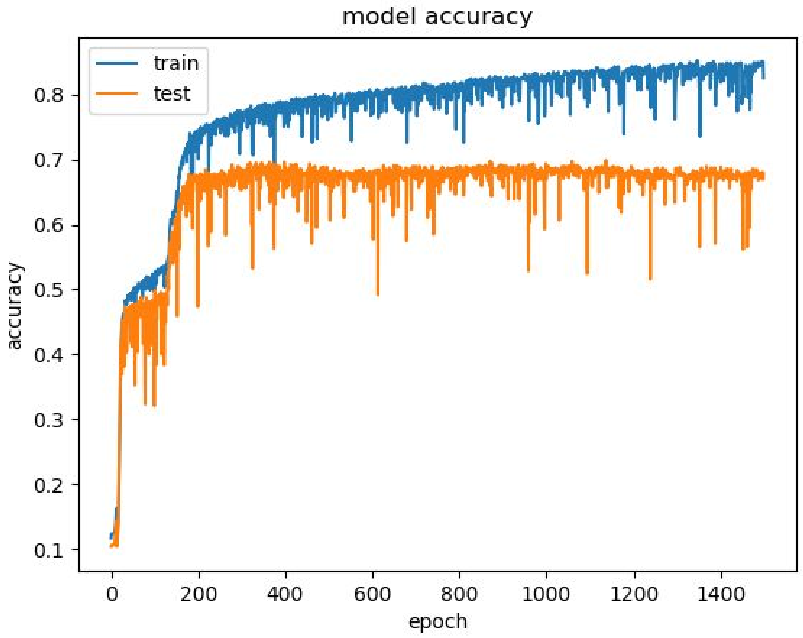
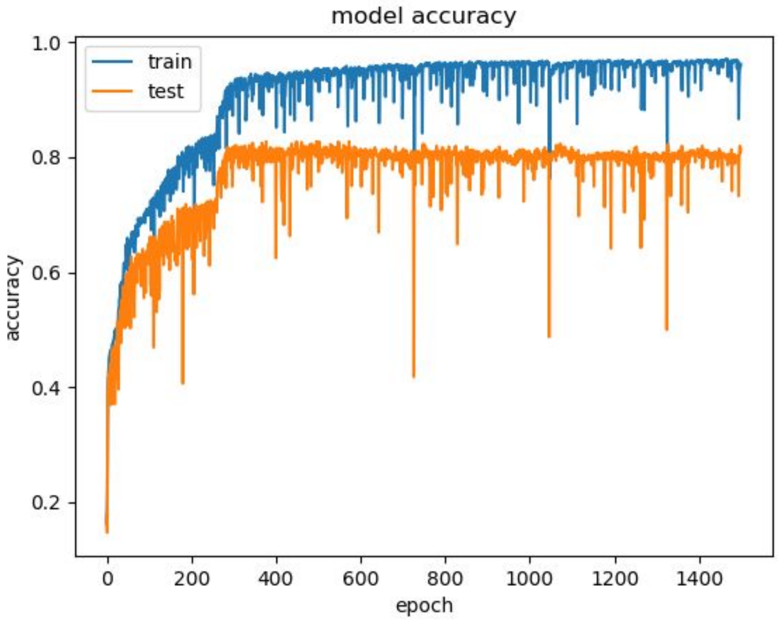
Two Hidden Layers: Max Validation Accuracy: 42.65%	
Three Hidden Layers: Max Validation Accuracy: 81.41%	

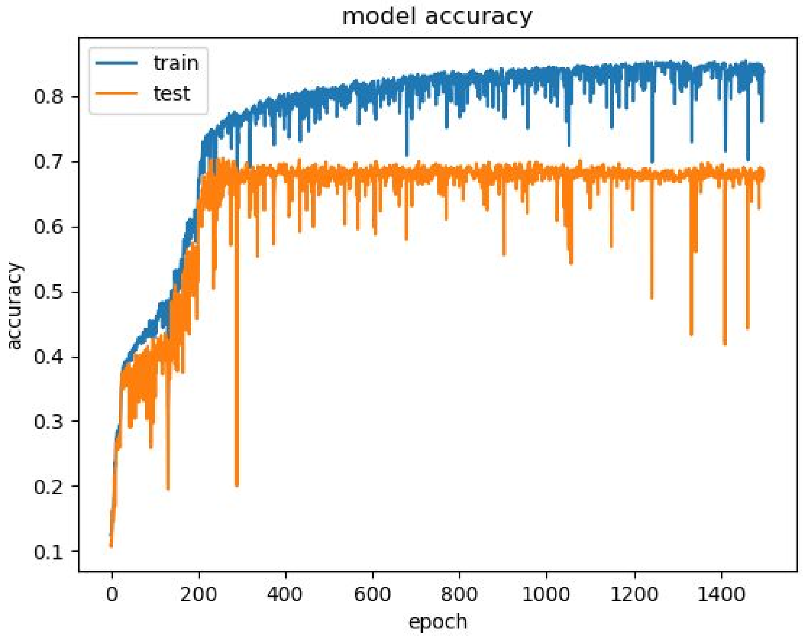
Summary of Hidden Layers:

The graph of two hidden layers is an example of how important initialization values were in achieving higher success rates, as seen in the figure, the prediction accuracy only reached 42.65% yet with just one hidden layer accuracy was 79.96% and with three it was 81.41%. There were only slight gains with the addition of the third hidden layer of 12 neurons.

Batch Size:

Setup: Epoch = 1500, Hidden Layers = 3

Batch Size = 128: Max Validation Accuracy: 67.82%	
Batch Size = 256: Max Validation Accuracy: 80.00%	

Batch Size = 512: Max Validation Accuracy: 68.33%	

Summary of Batch Size:

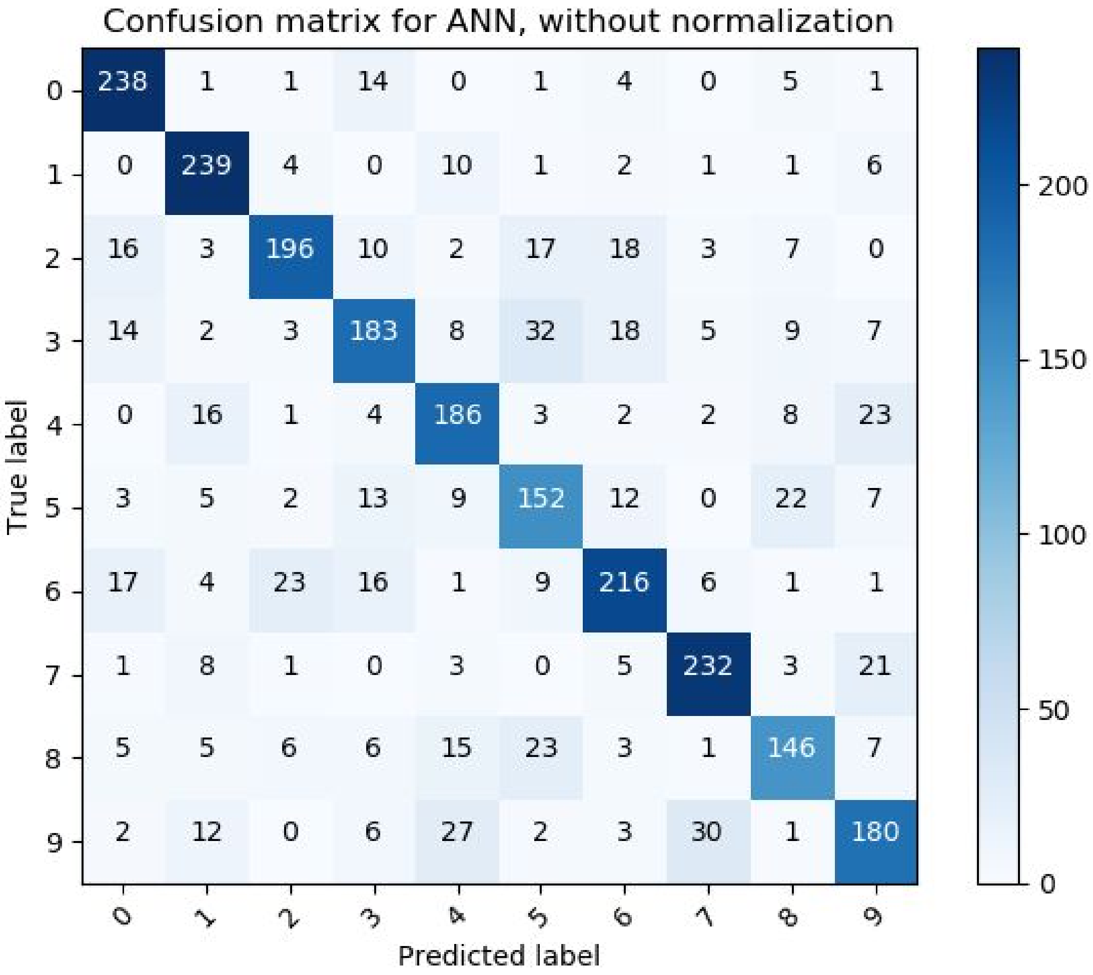
Although not displayed in graphs above, small batch size values < 64 were used, but never achieved an accuracy above 60%, Peak model accuracy seemed to be achieved with values between 150-300 with 256 performing the best. Batch size of 512 seemed two large and may have skipped over the goal loss minima for backpropagation.

Optimal Setup:

Hidden Layers = 3
Batch Size = 256
Epochs = 1500

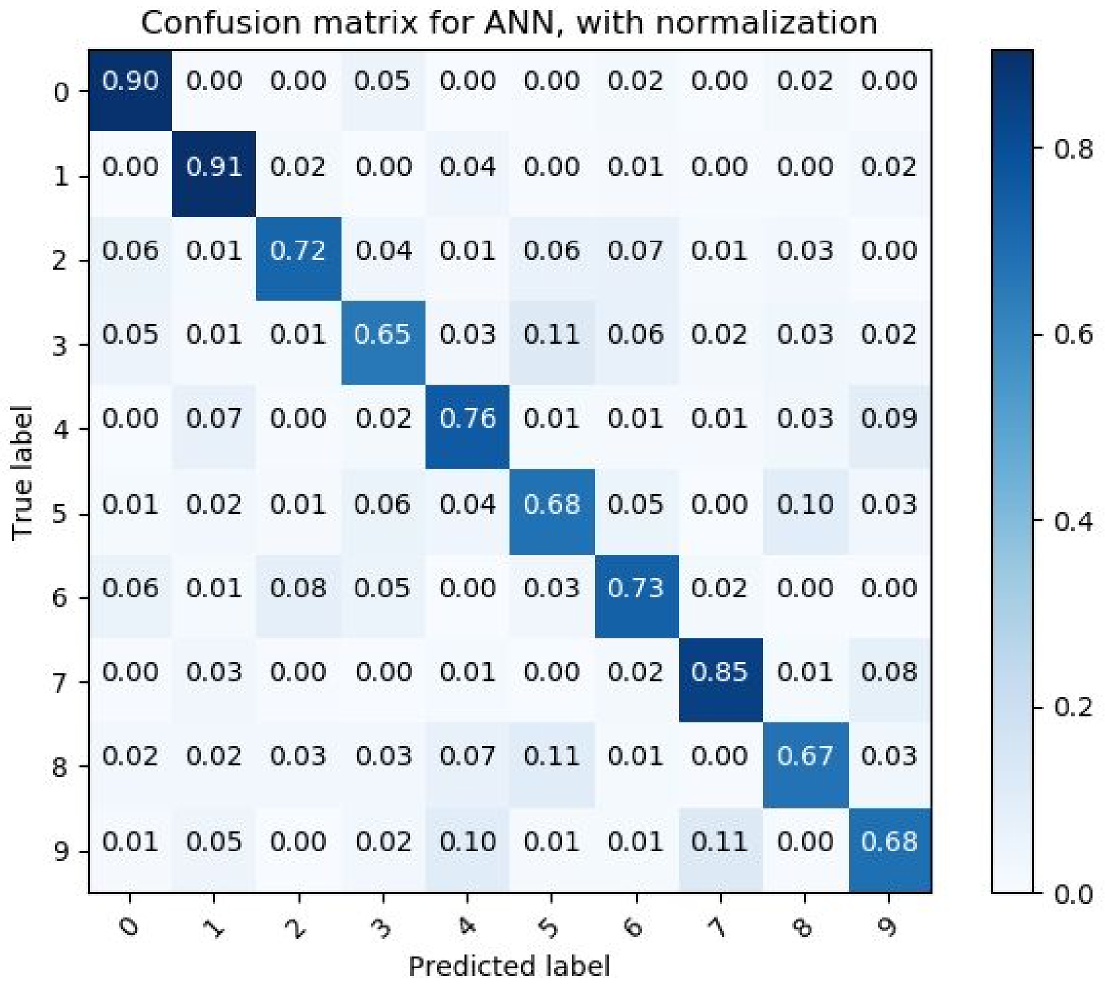
Results:

Below is the confusion matrix for the model where validation_split=0.2, epochs=1500 and batch_size=256:



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	238	1	1	14	0	1	4	0	5	1
1	0	239	4	0	10	1	2	1	1	6
2	16	3	196	10	2	17	18	3	7	0
3	14	2	3	183	8	32	18	5	9	7
4	0	16	1	4	186	3	2	2	8	23
5	3	5	2	13	9	152	12	0	22	7
6	17	4	23	16	1	9	216	6	1	1
7	1	8	1	0	3	0	5	232	3	21
8	5	5	6	6	15	23	3	1	146	7
9	2	12	0	6	27	2	3	30	1	180

Matplotlib view of non-normalized confusion matrix for artificial neural network.



	0	1	2	3	4	5	6	7	8	9
0	0.90	0.00	0.00	0.05	0.00	0.00	0.02	0.00	0.02	0.00
1	0.00	0.91	0.02	0.00	0.04	0.00	0.01	0.00	0.00	0.02
2	0.06	0.01	0.72	0.04	0.01	0.06	0.07	0.01	0.03	0.00
3	0.05	0.01	0.01	0.65	0.03	0.11	0.06	0.02	0.03	0.02
4	0.00	0.07	0.00	0.02	0.76	0.01	0.01	0.01	0.03	0.09
5	0.01	0.02	0.01	0.06	0.04	0.68	0.05	0.00	0.10	0.03
6	0.06	0.01	0.08	0.05	0.00	0.03	0.73	0.02	0.00	0.00
7	0.00	0.03	0.00	0.00	0.01	0.00	0.02	0.85	0.01	0.08
8	0.02	0.02	0.03	0.03	0.07	0.11	0.01	0.00	0.67	0.03
9	0.01	0.05	0.00	0.02	0.10	0.01	0.01	0.11	0.00	0.68

Matplotlib view of normalized confusion matrix for artificial neural network.

The total number of correctly predicted labels for this model was 1968 out of 2600. Thus, the Model Classifier Accuracy was 75.69%. The Model Classifier Error was 24.31%. The most accurate label was label=1 (91%), and the least accurate label was label=3 (63%).

Label:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Precision:	80.4	81.0	82.7	72.6	71.3	63.3	76.3	82.9	71.9	71.1
Recall:	89.8	90.5	72.1	65.1	75.9	67.6	73.5	84.7	67.3	68.4

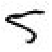
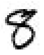
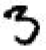
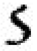
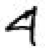
Precision (%) and Recall (%) for each label of the model where epochs=1500.

The label 7 had the greatest precision (82.9%) and label 1 had the greatest recall (90.5%). Therefore, few images were confused with the label 7, and few images with the attribute label=1 were misclassified as other labels. Labels 2, 1 and 0 also had high precision (82.7%, 81.0% and 80.4% respectively), and labels 0 and 7 also had high recall (89.8% and

84.7%). Many of the labels had low recall, with label 3 having the lowest (65.1%) and labels 8 and 4 also having low recall (67.3% and 67.6%). Therefore, many of the images with these actual labels were predicted as having different labels. The only outlier for low precision was label 5 (63.3%), so many images were misclassified as label=5.

Misclassification:

From the confusion matrix the most common mixups were between (5 and 8), (3 and 5) and (9 and 4)

Misclassified Image	Example of Predicted Image	Explanation
		Thought a 5 was an 8. The 5 itself is rather ambiguous, the curves of it are very similar to that of the 8. Curve detection seems to be an essential part of number detection.
		Thought 3 was a 5. This mistake is reasonable considering the 3 could be seen as an unfinished 5. The bottom part of the 3 resembles the bottom of the 5 quite closely.
		Thought 9 was a 4. Yea even I would say that's a four. Can't blame it here.

Decision Tree:

Baseline Model:

The Decision Tree baseline model was created using the *DecisionTreeClassifier* function from the scikit-learn Python package. The baseline tree was built using the function's default parameters. Scikit-learn's *fit* function was used to fit the training images and labels together. For the baseline model, the model's sole feature was the pixel map.

Following the fitting, the tree was run on the validation set of images and its accuracy was scored. The validation subset resulted in an accuracy of 77%.

Variations on the Baseline Model:

The baseline model's accuracy was already fairly good. Attempts were made to improve it through modifying the parameters for the *DecisionTreeClasifier* call. In an attempt to lessen generalization error via implementing pruning, the following parameters were changed: the max depth was limited to 15, the minimum samples required for a leaf node was raised to 2, and the splitter was set to "random." This resulted in a very slightly lower accuracy score for the validation subset: 75.8%. The accuracy score for the test subset was 75.7%.

The model was further modified to feature a slightly higher max depth, the default minimum samples for a leaf node (one), and a non-random splitter. This resulted in a miniscule improvement.

Ultimately, we settled on a max depth of 19 and a non-default criterion. The criterion was set to “entropy,” thus prioritizing information gain over the gini impurity. Due to the criterion change, the accuracy score for the validation subset was 77.6%. The accuracy score for the test subset was 77.8%. We settled on this for the baseline model variation.

Hand-Engineered Features

Six discriminatory features were extracted from the images’ pixel arrays. The decision tree was then trained on these new features exclusively.

The first of the six features was the average pixel density. The total “occupied” pixels were divided by the total possible pixels in a given image. The primary reasoning behind this feature was to allow the tree to discern between numbers that took up less space and those that took up more. For instance, “1” should take up less space than an “8.”

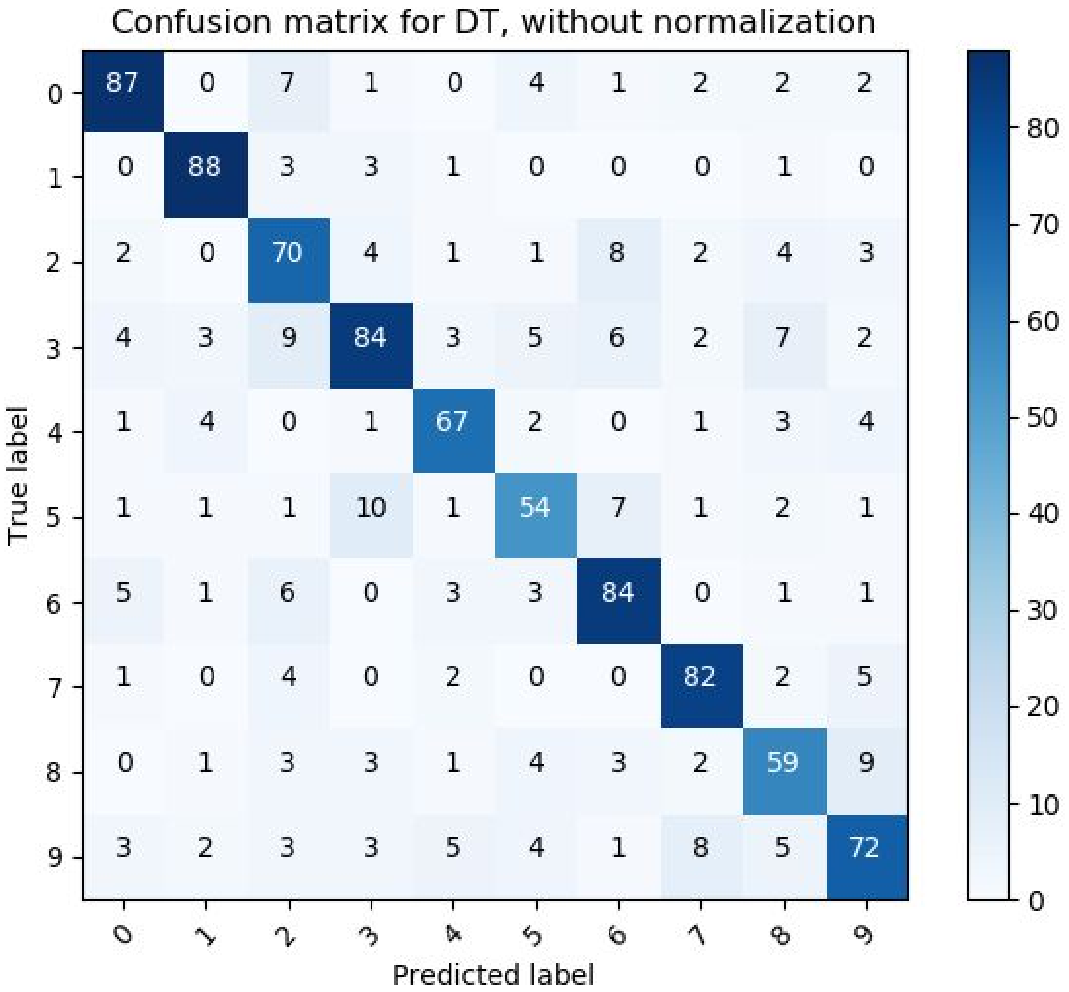
The second of the six was the total amount of blank pixels in a given image. This feature accomplishes a similar thing as the first, but in a slightly different way. Again, numbers such as “1” or “7” should have more blank pixels than an a 6 or 8.

The third feature and fourth features were the highest and lowest pixel locations. These features were extracted via iterating through a given image’s pixel array and storing the index of the highest and lowest “occupied” pixel. These features served to be mostly superfluous, as most numbers have near-identical high and low locations.

The fifth and sixth features were the leftmost and rightmost pixel locations. These features were similar to the previous two, but proved to be slightly more useful, primarily in discerning the number “1” from the rest of the bunch.

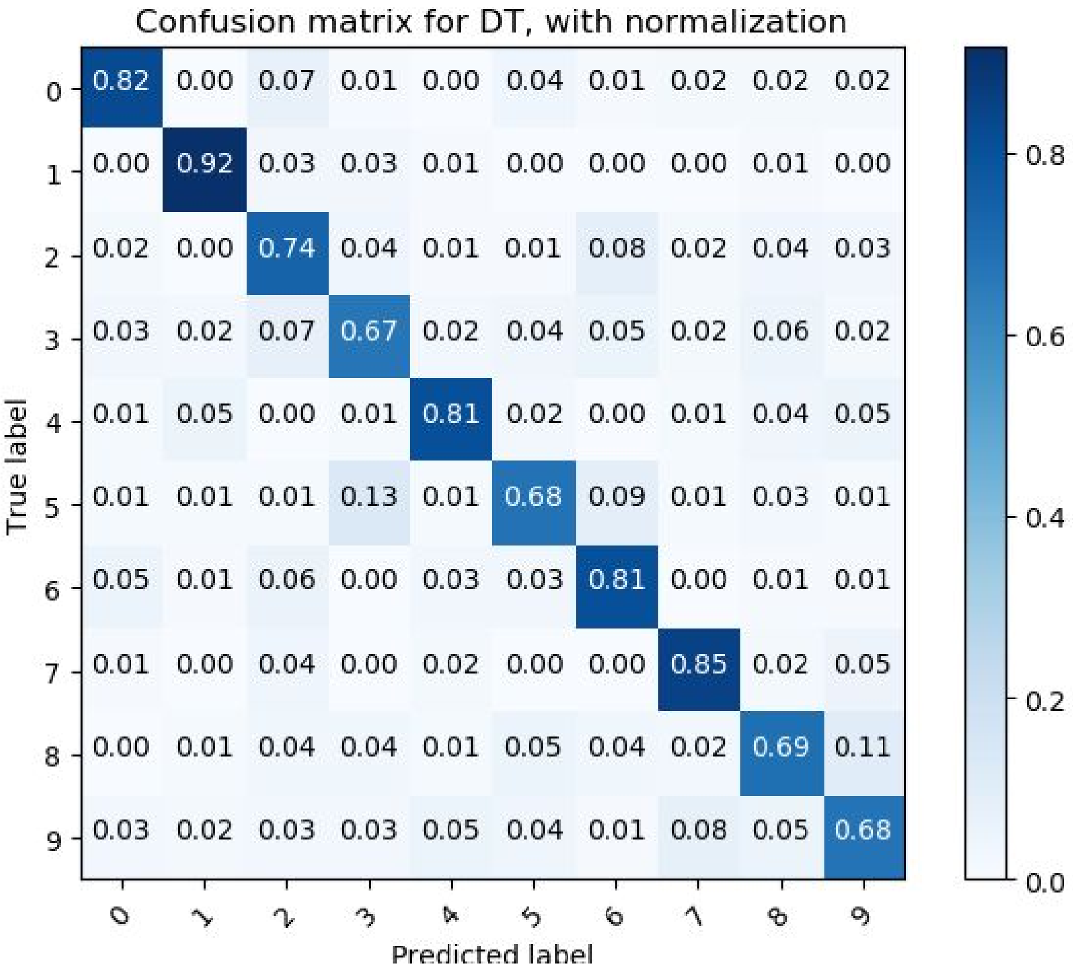
These hand-engineered features proved to be vastly inferior to using the overall pixel array. When trained on these features, the decision tree was only able to accomplish an accuracy score of 49.2% on the validation subset. The accuracy score of the test subset was likewise much lower at 48.9%.

Results:



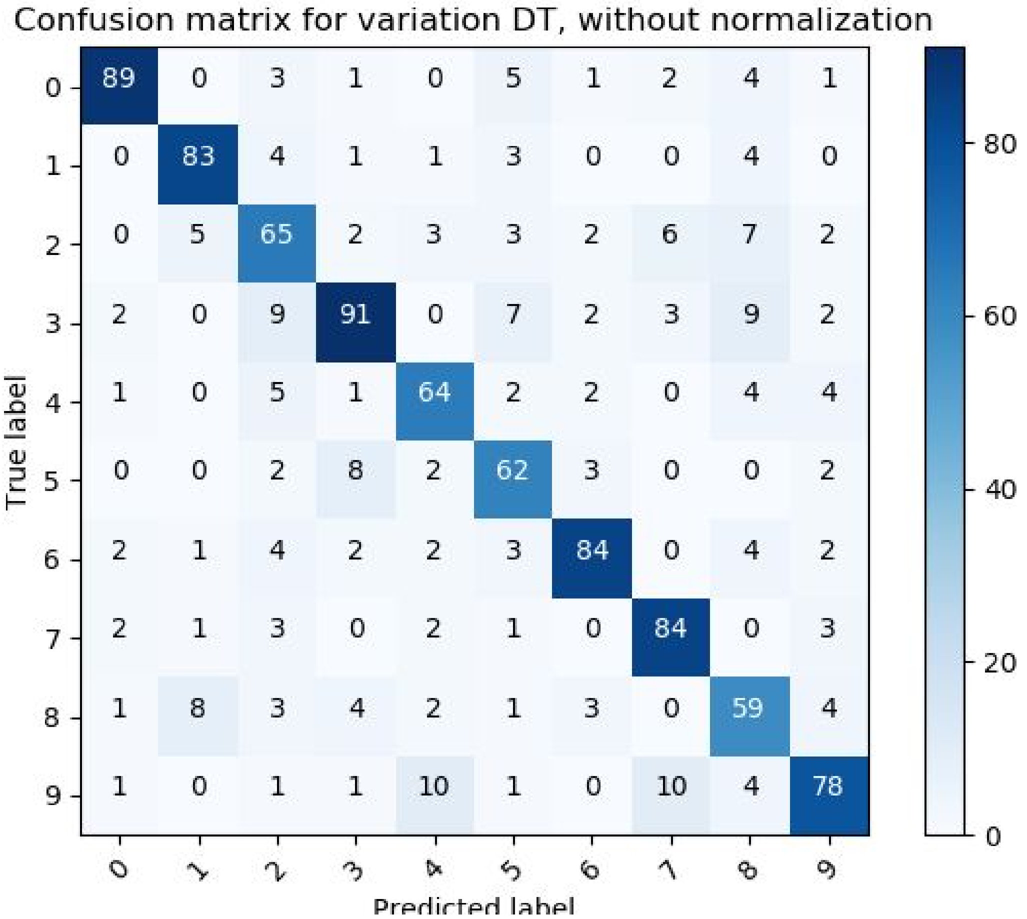
	0	1	2	3	4	5	6	7	8	9
0	87	0	7	1	0	4	1	2	2	2
1	0	88	3	3	1	0	0	0	1	0
2	2	0	70	4	1	1	8	2	4	3
3	4	3	9	84	3	5	6	2	7	2
4	1	4	0	1	67	2	0	1	3	4
5	1	1	1	10	1	54	7	1	2	1
6	5	1	6	0	3	3	84	0	1	1
7	1	0	4	0	2	0	0	82	2	5
8	0	1	3	3	1	4	3	2	59	9
9	3	2	3	3	5	4	1	8	5	72

Matplotlib view of non-normalized confusion matrix for baseline model using validation set.



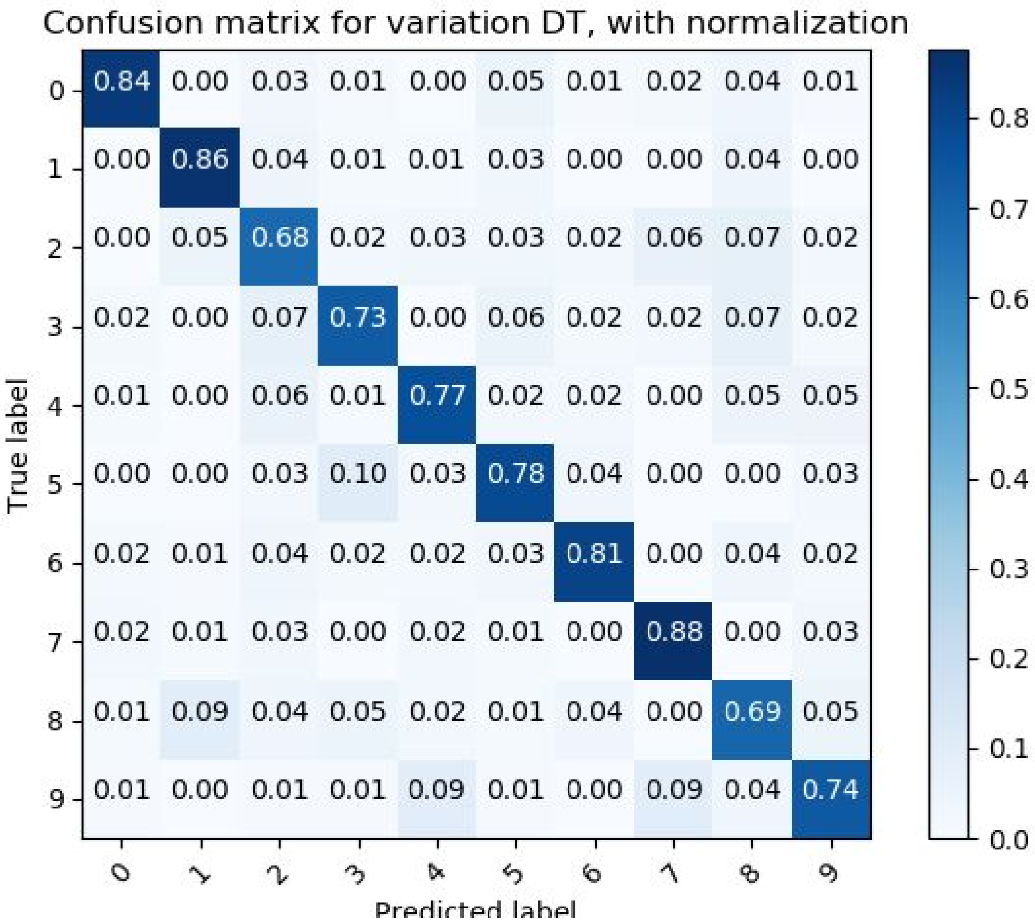
True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	0.82	0.00	0.07	0.01	0.00	0.04	0.01	0.02	0.02	0.02
1	0.00	0.92	0.03	0.03	0.01	0.00	0.00	0.00	0.01	0.00
2	0.02	0.00	0.74	0.04	0.01	0.01	0.08	0.02	0.04	0.03
3	0.03	0.02	0.07	0.67	0.02	0.04	0.05	0.02	0.06	0.02
4	0.01	0.05	0.00	0.01	0.81	0.02	0.00	0.01	0.04	0.05
5	0.01	0.01	0.01	0.13	0.01	0.68	0.09	0.01	0.03	0.01
6	0.05	0.01	0.06	0.00	0.03	0.03	0.81	0.00	0.01	0.01
7	0.01	0.00	0.04	0.00	0.02	0.00	0.00	0.85	0.02	0.05
8	0.00	0.01	0.04	0.04	0.01	0.05	0.04	0.02	0.69	0.11
9	0.03	0.02	0.03	0.03	0.05	0.04	0.01	0.08	0.05	0.68

Matplotlib view of normalized confusion matrix for baseline model using validation set.



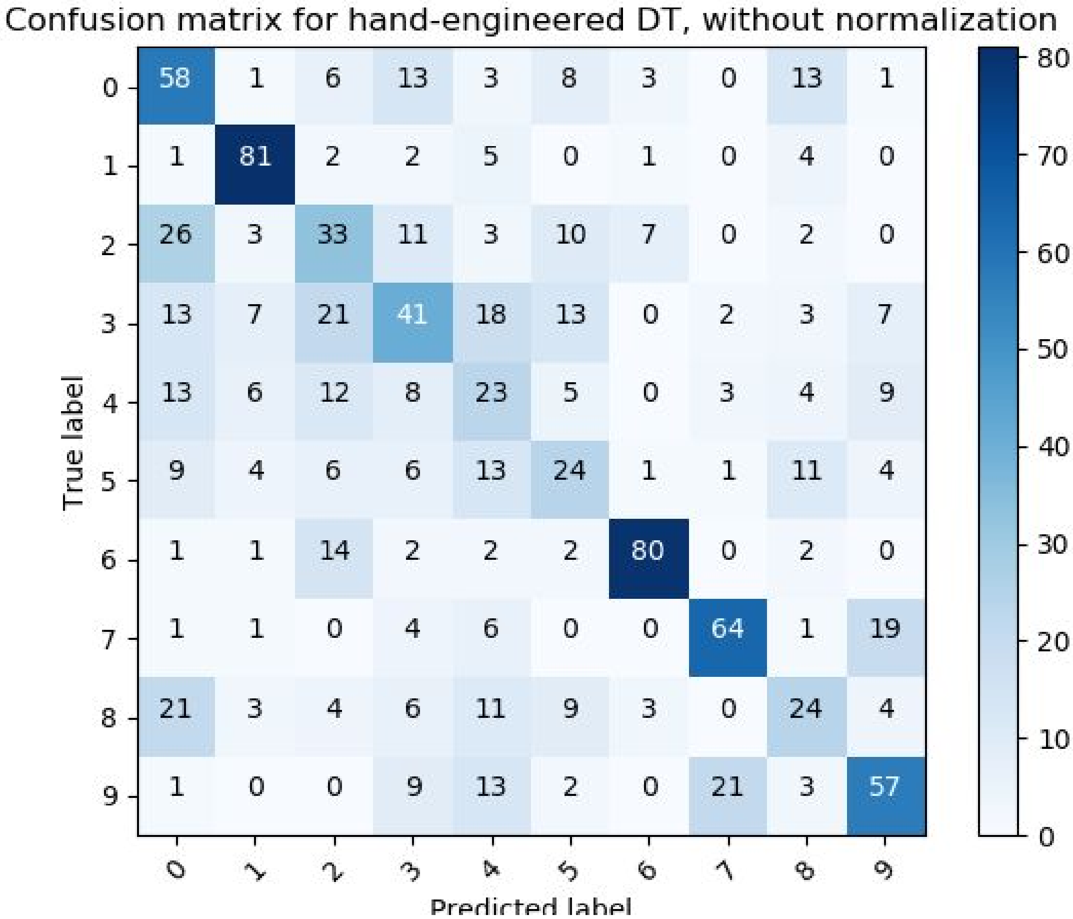
	0	1	2	3	4	5	6	7	8	9
0	89	0	3	1	0	5	1	2	4	1
1	0	83	4	1	1	3	0	0	4	0
2	0	5	65	2	3	3	2	6	7	2
3	2	0	9	91	0	7	2	3	9	2
4	1	0	5	1	64	2	2	0	4	4
5	0	0	2	8	2	62	3	0	0	2
6	2	1	4	2	2	3	84	0	4	2
7	2	1	3	0	2	1	0	84	0	3
8	1	8	3	4	2	1	3	0	59	4
9	1	0	1	1	10	1	0	10	4	78

Matplotlib view of non-normalized confusion matrix for variation model using validation set.



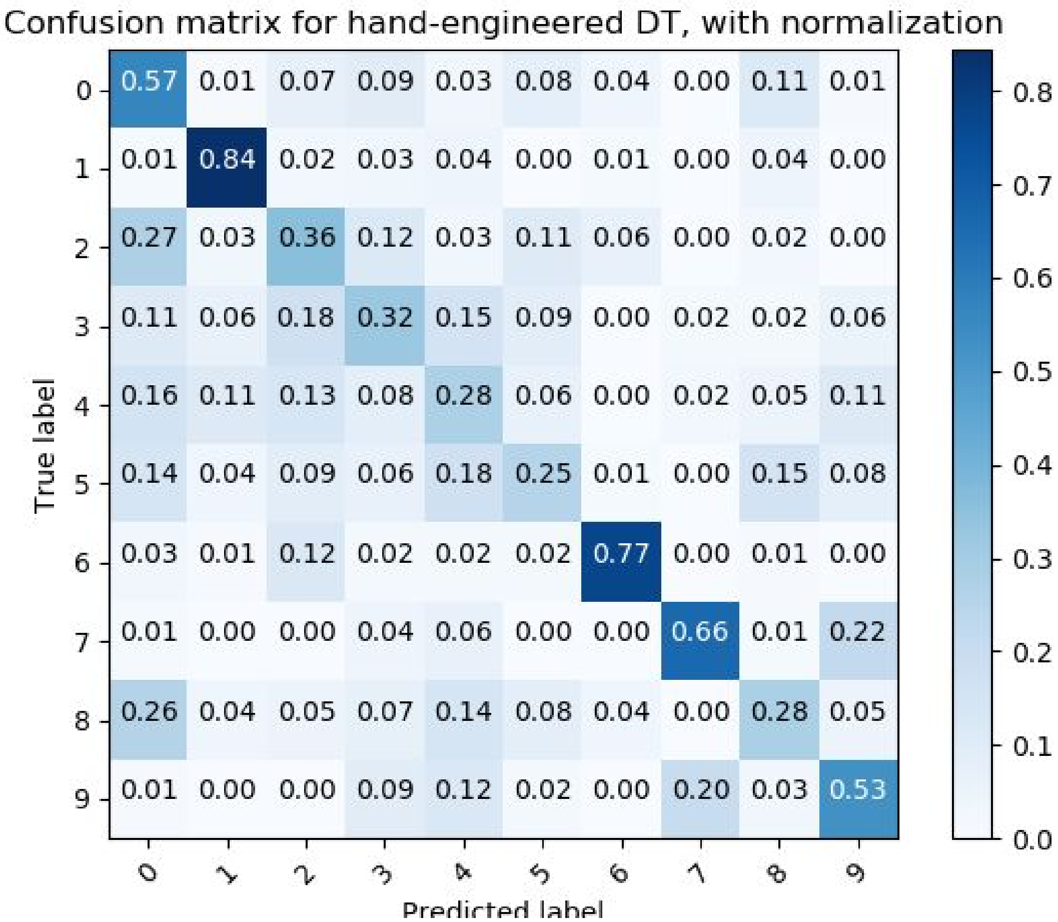
True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	0.84	0.00	0.03	0.01	0.00	0.05	0.01	0.02	0.04	0.01
1	0.00	0.86	0.04	0.01	0.01	0.03	0.00	0.00	0.04	0.00
2	0.00	0.05	0.68	0.02	0.03	0.03	0.02	0.06	0.07	0.02
3	0.02	0.00	0.07	0.73	0.00	0.06	0.02	0.02	0.07	0.02
4	0.01	0.00	0.06	0.01	0.77	0.02	0.02	0.00	0.05	0.05
5	0.00	0.00	0.03	0.10	0.03	0.78	0.04	0.00	0.00	0.03
6	0.02	0.01	0.04	0.02	0.02	0.03	0.81	0.00	0.04	0.02
7	0.02	0.01	0.03	0.00	0.02	0.01	0.00	0.88	0.00	0.03
8	0.01	0.09	0.04	0.05	0.02	0.01	0.04	0.00	0.69	0.05
9	0.01	0.00	0.01	0.01	0.09	0.01	0.00	0.09	0.04	0.74

Matplotlib view of normalized confusion matrix for variation model using validation set.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	58	1	6	13	3	8	3	0	13	1
1	1	81	2	2	5	0	1	0	4	0
2	26	3	33	11	3	10	7	0	2	0
3	13	7	21	41	18	13	0	2	3	7
4	13	6	12	8	23	5	0	3	4	9
5	9	4	6	6	13	24	1	1	11	4
6	1	1	14	2	2	2	80	0	2	0
7	1	1	0	4	6	0	0	64	1	19
8	21	3	4	6	11	9	3	0	24	4
9	1	0	0	9	13	2	0	21	3	57

Matplotlib view of non-normalized confusion matrix for hand-done model using validation set.



Matplotlib view of normalized confusion matrix for hand-done model using validation set.

The total number of positive results was 750 out of 975 test images for the baseline model. Therefore, the Model Classifier Accuracy for this baseline model is 76.92%. The Model Classifier Error is 23.07%. Label 1 yielded the most positive results at 92% accuracy, and label 3 yielded the least positive results at 67% accuracy.

The total positive results for the baseline variation model was 759 out of 975, resulting in a Model Classifier Accuracy of 77.8%. THe Model Classifier Error was thus 22.2%. The label with the highest accuracy was label=7 (88%) and the label with the lowest accuracy was label=2 (68%).

The hand-engineered model had the lowest Model Classifier Accuracy (485 out of 975, 49.7%) and the highest Model Classifier Accuracy (50.3%). The most accurate label was label=1 (84%) and the least accurate label was label=5 (25%).

Label:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Precision:	84.5	82.9	68.3	79.2	75.3	74.4	80.9	81.2	63.2	76.0
Recall:	82.1	84.4	72.6	67.2	80.7	73.4	81.7	85.4	64.7	71.7

Precision (%) and Recall (%) for each label of the baseline model.

Label 0 for the baseline model had the greatest precision (84.5%), so few of the images were misclassified as label=0. Labels 0, 1 and 7 also had fairly high precisions (84.5%, 82.9% and 81.2%, respectively). The greatest recall out of the labels was label 7 (85.4%), so for the most part test images with an associated label of 7 were classified correctly. Labels 1, 0 and 6 also had high recall percentages (84.4%, 82.1%, 81.7%, respectively). Label 8 had both the lowest precision (63.2%) and lowest recall (64.7%), so just under half of images classified as label=8 were classified incorrectly, and just over half of test images with an associated label of 8 were classified correctly.

Label:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Precision:	92.5	82.2	67.3	80.2	74.1	74.4	84.7	77.7	61.7	79.8
Recall:	81.1	86.5	71.6	71.2	75.9	77.2	79.8	90.6	68.2	78.3

Precision (%) and Recall (%) for each label of the variation model.

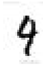
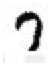
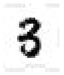
Label 1 for the variation model has the highest precision (92.5%). Thus, very few images were misclassified as being number 0. Labels 1, 3, and 7 also had fairly high precisions (82.2%, 80.2%, and 84.7%, respectively). The greatest recall label was for the number 7 (90.6). Much like the baseline model, this means that most values that were classified as 7 were classified correctly. Previously low precision values (2 and 8) have remained low, as did previously low recall values. Additionally, the highest precision values increased from the baseline to the variation model. Specifically, 6, 0, and 3 increased by at least a full percent. Some values lowered, such as 1 and 4.

Label:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Precision:	39.5	74.3	34.0	40.8	23.2	30.8	84.2	71.6	36.9	53.8
Recall:	56.6	84.4	35.8	32.0	27.7	25.3	76.9	65.6	28.2	52.8

Precision (%) and Recall (%) for each label of the hand-engineered model.

Compared to the variation model, the hand-engineered model performed worse across-the-board. Most precision labels were significantly lower, though there are some minor exceptions. The number 6, for instance, maintained a fairly high precision value (84.2%). However, not a single precision value increased between the variation and the hand-engineered model. However, the recall values also decreased significantly. However, previously high recall values maintained fairly high. 1, 6, and 7 still carry the highest recall values (84,4%, 76,9%, and 65.6%, respectively). Therefore, the decision tree mislabeled many labels initially yet never corrected itself.

Visualization
Based on the confusion matrix, the most common misclassifications for the variation model were 4 and 9, 7 and 9, and 3 and 8.

Misclassified Image	Example of Predicted Image	Explanation
		4’s are visually very similar to 9’s. They both feature rounded tops and narrow bottoms.
		This particular 7 is very close to a 9 due to its rounded top and narrow bottom.
		This particular 3 is very close to an 8 due to its upper and lower curves nearly touching.

Comparison to Neural Networks

The neural networks achieved a maximum accuracy of 81.41%. Comparatively, the decision tree variation model achieved a maximum accuracy of 77.8%. Thus, the neural network performed significantly better than the best decision tree model. However, the neural network’s lack of consistency due to the random initialization could potentially result in the decision tree performing higher.

K-Nearest:

Experimentation:

To create the K-Nearest algorithm, *KNeighborsClassifier* was used from the scikit-learn package in Python. The “k” input value for this function is denoted as the n_neighbors option, and the initial value that was fed in was 3. The *fit* function from scikit-learn was used to fit the training images and training labels together. The predict function was used on the result of the fit with the testing set to make sure that everything worked properly. Lastly, we fed the prediction values into the *confusion_matrix* function to produce a confusion matrix of the predicted labels versus the actual labels.

Once we were satisfied the algorithm work, we began testing different values of k against the validation set. We made an automated function that tested ranges of values for k, and chose the best k to test with the test set. We used values K_LOW and K_HIGH, where

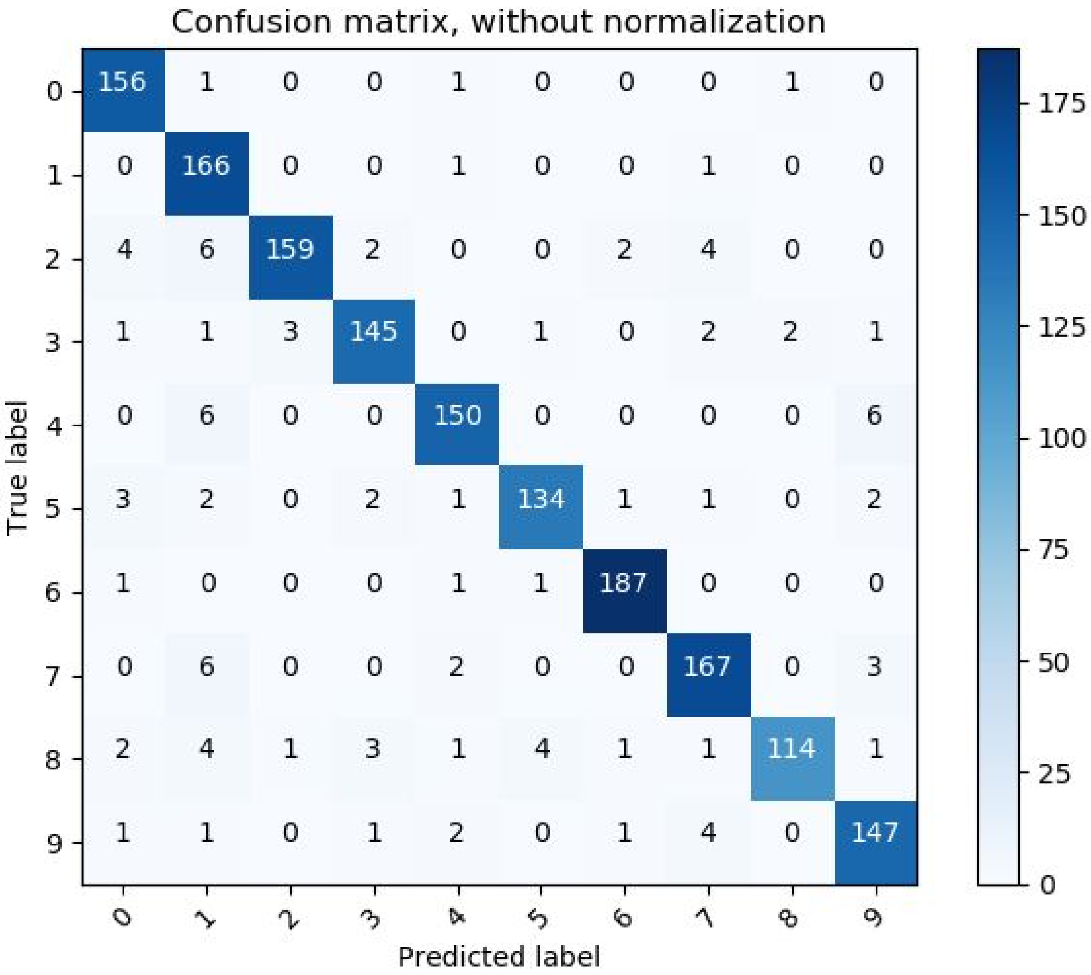
K_LOW was the start of the range, and K_HIGH was the end of the range. For each k that was tested against the validation set, we printed a confusion matrix to see what difference, if any, occurred.

By looking at the different confusion matrices that were produced using the validation set, we determined that there were minor differences between different values of k. The diagonal values stayed relatively the same, with minor increments or decrements of one or two between the different k values. The negative test values in the matrix varied much more with different values of k, ranging from 0 to 10, with some outliers being higher. When testing k in a range from 1 to 10 against the validation data, the best k value was 1. When testing k in a range from 1 to 50, the best k value was still 1. With the K-Nearest function that was created, this k is automatically fed to be tested against the test set.

Even though the k value of 1 was the lowest possible value in both of the above ranges, lower values of k would not necessarily yield the most accurate results. For example, when testing k in a range from 5 to 50, the best k value was 6. When testing k in a range from 30 to 50, the best k value was 31. These values are still some of the lower values, but not the lowest value of each range. Regardless, it is clear that lower values of k are preferred with how our data sets are split.

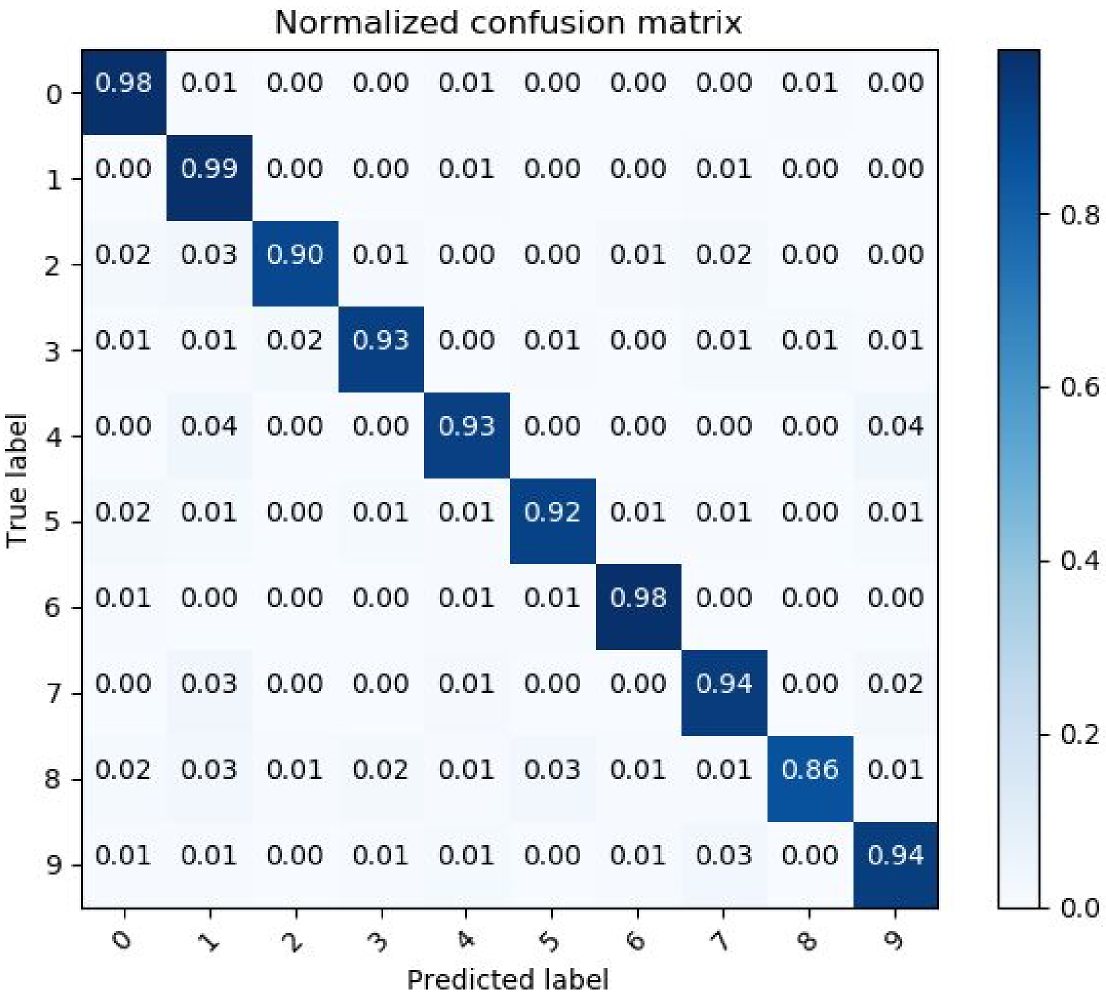
<u>Results:</u>

With a k value of 1, we got the resulting confusion matrix:



	0	1	2	3	4	5	6	7	8	9
0	156	1	0	0	1	0	0	0	1	0
1	0	166	0	0	1	0	0	1	0	0
2	4	6	159	2	0	0	2	4	0	0
3	1	1	3	145	0	1	0	2	2	1
4	0	6	0	0	150	0	0	0	0	6
5	3	2	0	2	1	134	1	1	0	2
6	1	0	0	0	1	1	187	0	0	0
7	0	6	0	0	2	0	0	167	0	3
8	2	4	1	3	1	4	1	1	114	1
9	1	1	0	1	2	0	1	4	0	147

Matplotlib view of non-normalized confusion matrix for k = 1.



Matplotlib view of normalized confusion matrix for k = 1.

The total number of positive results was 1525 out of the total 1625 test images. The Model Classifier Accuracy for this model is thus 93.84%, with the label 1 yielding the most positive results at 99% accuracy and the label 8 yielding the least amount of positive results at 86% accuracy. The Model Classification Error is 6.15%, with the label 8 yielding the highest error percent at 14%.

Label:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Precision:	92.9	86.0	97.6	94.8	94.3	95.7	97.7	92.8	97.4	91.9
Recall:	98.1	98.8	89.8	92.9	92.6	91.8	98.4	93.8	86.4	93.6

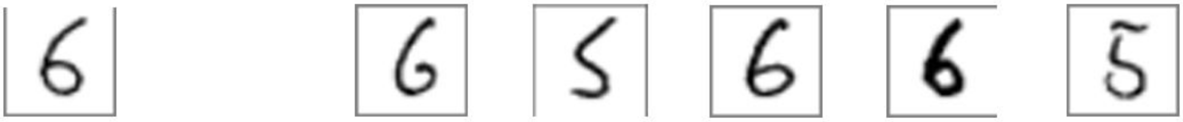
Precision (%) and Recall (%) for each label of the model where k=1.

Label 6 had the greatest precision (97.7%) and recall (98.4%), which means few images were confused with label=6 and few images with label=6 were confused with other labels.

Labels 2 and 8 were also had high precision (97.6% and 97.4% respectively), however label 8 had the lowest recall (86.4%). Thus, although few images with other labels were confused with label=8, the highest amount of confused images were images where label=8 was misclassified. Label 1 had the lowest precision (86%), so the label that was most misclassified under was 1.

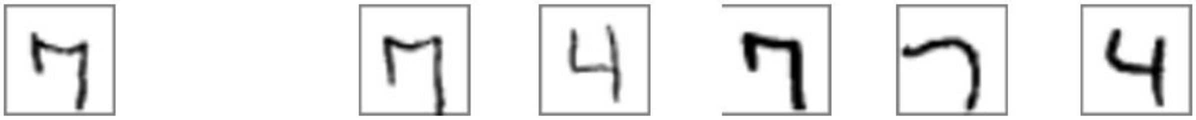
Misclassifications:

Below are some misclassified images from the test set and their five closest neighbors (k=5). Keep in mind that neighbors may not be in order of closeness.



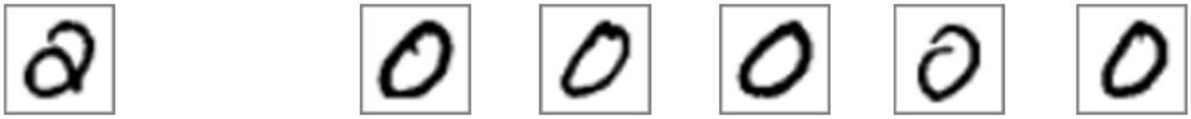
Misclassified 6 (left) with its five closest neighbors (right) for k-nearest neighbors.

For this image, the predicted label was classified as 5, but the actual label was 6. Of the five neighbors, two of them are classified as 5, and at least for the left-most 5 there is a distinct similarity to the tested image. This 5 has a similar appearance to the other 6's in that the top is sloped down to the left and the bottom looks like the loop at the bottom of the 6. In this case it seems that the misclassification was not due to the test image itself, but due to the way that specific 5's were written. A human would likely not misclassify this test image as a 5, but it is possible that a human could misclassify the left-most 5 as a 6.



Misclassified 7 (left) with its five closest neighbors (right) for k-nearest neighbors.

In this second case, the predicted label was classified as 4, when in actuality the label was a 7. Again of the five neighbors two of them are classified as 4 while the remaining are classified as the actual label. It is less apparent for this case why the image was misclassified, since the test image does not necessarily look similar to the two 4's represented by its neighbors. It could be because of the vertical line at the left side of the test image, and the fact that both this test image and the represented 4's all have two vertical and one horizontal line. A human would likely not misclassify this test image as a 4.



Misclassified 2 (left) with its five closest neighbors (right) for k-nearest neighbors.

In this last case of misclassification, it is the most clear why the misclassification took place. The predicted label was 0, but the actual label is 2, and all five closest neighbors to the test image are classified as 0. This test image looks very similar to the neighboring images due to the looping bottom which makes the image look circular. It is conceivable that a human could misclassify the test image as a 0 because of how it appears.

Comparison to Decision Trees and Neural Networks

Compared to the decision tree's highest accuracy of 77.8% and the neural networks' accuracy of 81.41%, the K-Nearest algorithm's accuracy was 93.84%. This accuracy is significantly higher than the other two algorithms. Additionally, K-Nearest's recall scores were significantly (typically 10% or more) higher than those for the decision tree. Thus, K-Nearest resulted in far more corrections to what would otherwise be misclassifications.