1. a)

z_1 is a $d \times 1$ vector because $z_1 = \sigma^{-1}(h)$ where $h \in \mathbf{R}^d$.

$W^{(1)}$ is a $d \times d$ matrix because $x \in \mathbf{R}^d$ and $W^{(1)}x = z_1 \in \mathbf{R}^d$.

z_2 is a $d \times 1$ vector because it is a sum of two d-dimensional vectors.

$W^{(2)}$ is a $1 \times d$ vector because $W^{(2)}z_2 = y \in \mathbf{R}$.

b)

Parameters are the things that are learned through training, and the only things learned through training in this model is $W^{(1)}$ and $W^{(2)}$, thus the number of parameters is equal to the dimension of $W^{(1)}$ plus the dimension of $W^{(2)}$, which is $d^2 + d$.

c)

$$\overline{y} = \frac{\partial L}{\partial y} = y - t$$

$$\begin{aligned}\overline{W^{(2)}} &= \frac{\partial L}{\partial y}\frac{\partial y}{\partial W^{(2)}} = \overline{y} z_2^T \\ &= (y - t) z_2^T\end{aligned}$$

$$\begin{aligned}\overline{z_2} &= \frac{\partial L}{\partial y}\frac{\partial y}{\partial z_2} = W^{(2)}\overline{y} \\ &= W^{(2)}(y - t)\end{aligned}$$

$$\begin{aligned}\overline{h} &= \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial h} = \overline{z_2} \\ &= W^{(2)}(y - t)\end{aligned}$$

$$\begin{aligned}\overline{z_1} &= \frac{\partial L}{\partial h}\frac{\partial h}{\partial z_1} = \overline{h} \circ \sigma^{-1}(z_1) \\ &= (W^{(2)}(y - t)) \circ \sigma^{-1}(z_1)\end{aligned}$$

$$\begin{aligned}\overline{W^{(1)}} &= \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial W^{(1)}} = \overline{z_1} x^T \\ &= ((W^{(2)}(y - t)) \circ \sigma^{-1}(z_1)) x^T\end{aligned}$$

$$\begin{aligned}\overline{x} &= \frac{\partial L}{\partial z_1}\frac{\partial z_1}{\partial x} + \frac{\partial L}{\partial z_2}\frac{\partial z_2}{\partial x} = W^{(1)}\overline{z_1} + \overline{z_2} \\ &= W^{(1)}(W^{(2)}(y - t)) \circ \sigma^{-1}(z_1) + W^{(2)}(y - t)\end{aligned}$$

2. a)

Let $f(z_k) = \exp(z_k)$ and $g(z_{k'}) = \sum_{k'=1}^{K} exp(z_{k'})$, then there are two cases for computing $\frac{\partial y_k}{\partial z_{k'}}$:

Case 1: $k \neq k'$

$$f'(z_k) = \frac{\partial f(z_k)}{\partial z_{k'}} = 0$$

$$g'(z_{k'}) = \frac{\partial g(z_{k'})}{\partial z_{k'}} = \exp(z_{k'})$$

$$\frac{\partial y_k}{\partial z_{k'}} = \frac{g(z_{k'})f'(z_k) - g'(z_{k'})f(z_k)}{g(z_{k'})^2} = -\frac{exp(z_{k'})exp(z_k)}{g(z_{k'})^2}$$

$$= -\frac{exp(z_{k'})}{g(z_{k'})}\frac{exp(z_k)}{g(z_{k'})} = -\frac{f(z_{k'})}{g(z_{k'})}\frac{f(z_k)}{g(z_{k'})}$$

$$= -y_k y_{k'}$$

Case 2: k = k'

$$f'(z_k) = \frac{\partial f(z_k)}{\partial z_k} = \exp(z_k)$$

$$g'(z_{k'}) = \frac{\partial g(z_{k'})}{\partial z_k} = \exp(z_k)$$

$$\frac{\partial y_k}{\partial z_{k'}} = \frac{g(z_{k'})f'(z_k) - g'(z_{k'})f(z_k)}{g(z_{k'})^2} = \frac{exp(z_k)g(z_{k'}) - exp(z_k)exp(z_k)}{g(z_{k'})^2}$$

$$= \frac{exp(z_k)g(z_{k'})}{g(z_{k'})^2} - \frac{exp(z_k)exp(z_k)}{g(z_{k'})^2}$$

$$= \frac{exp(z_k)}{g(z_{k'})} - \frac{exp(z_k)}{g(z_{k'})}\frac{exp(z_k)}{g(z_{k'})}$$

$$= \frac{f(z_k)}{g(z_{k'})} - \frac{f(z_k)}{g(z_{k'})}\frac{f(z_k)}{g(z_{k'})}$$

$$= y_k - y_k^2$$

b)

$$\frac{\partial L_{CE}}{\partial W_k} = \frac{\partial L_{CE}}{\partial y_k}\frac{\partial y_k}{\partial z_k}\frac{\partial z_k}{\partial W_k}$$

$$= -\frac{t_k}{y_k}\frac{\partial y_k}{\partial z_k}\frac{\partial z_k}{\partial W_k}$$

$$= -\frac{t_k}{y_k}(y_k - y_k^2)\frac{\partial z_k}{\partial W_k}$$ #By part a, $\frac{\partial y_k}{\partial z_k} = y_k - y_k^2$ when y and z has the same k

$$= -\frac{t_k}{y_k}(1 - y_k)y_k\frac{\partial z_k}{\partial W_k}$$

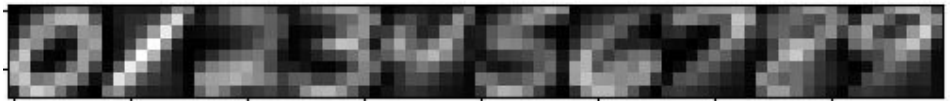
$$= -t_k(1 - y_k)\frac{\partial z_k}{\partial W_k}$$

$$= (y_k - t_k)\frac{\partial z_k}{\partial W_k}$$

$$= (y_k - t_k)x$$

3. 0)

Source codes for the figure are in q3_0.py and the generated picture is as the following:



1) Source codes are in q3_1.py.

a)

```
For k = 1, the train accuracy is 1.0 and the test accuracy is 0.96875
For k = 15, the train accuracy is 0.9571428571428572 and the test accuracy is 0.9535
```

b) For the situation of ties, the average Euclidean distance of the data points with the same label to the query point will be computed. The label with a lower average Euclidean distance to the query point will be selected as the output because a lower average Euclidean distance indicates a higher similarity to the query point.

c) The average validation accuracy for each value of k using 10 fold cross-validation is as the following:

```
The average accuracy for k = 1 using 10-Fold-Cross-Validationis: 0.964142857142857
The average accuracy for k = 2 using 10-Fold-Cross-Validationis: 0.9642857142857144
The average accuracy for k = 3 using 10-Fold-Cross-Validationis: 0.964142857142857
The average accuracy for k = 4 using 10-Fold-Cross-Validationis: 0.9639999999999999
The average accuracy for k = 5 using 10-Fold-Cross-Validationis: 0.962142857142857
The average accuracy for k = 6 using 10-Fold-Cross-Validationis: 0.9611428571428571
The average accuracy for k = 7 using 10-Fold-Cross-Validationis: 0.9582857142857142
The average accuracy for k = 8 using 10-Fold-Cross-Validationis: 0.9575714285714285
The average accuracy for k = 9 using 10-Fold-Cross-Validationis: 0.9532857142857143
The average accuracy for k = 10 using 10-Fold-Cross-Validationis: 0.9535714285714285
The average accuracy for k = 11 using 10-Fold-Cross-Validationis: 0.9514285714285714
The average accuracy for k = 12 using 10-Fold-Cross-Validationis: 0.9495714285714285
The average accuracy for k = 13 using 10-Fold-Cross-Validationis: 0.9480000000000001
The average accuracy for k = 14 using 10-Fold-Cross-Validationis: 0.9462857142857143
The average accuracy for k = 15 using 10-Fold-Cross-Validationis: 0.9438571428571428
```

From the graph, it is clear to see that the best accuracy is obtained when k equals 2, and the accuracy starts to decrease for k = 3 to 15. This result is reasonable because higher k leads to a tendency of overfitting, which reduces the accuracy.

2) For this part, the classification accuracy will be used as a basic performance measurement for comparing the Neural Network, SVM and AdaBoost. More performance metrics will be introduced in part 3.

a) Neural Network: q3_2_nn.py

The neural network in my implementation has 3 hidden layers. The first and second hidden layer has 64 neurons in each layer and the third hidden layer has 10 neurons. The sigmoid function was used as an activation function for all of the 3 hidden layers, and the softmax function was used for the output layer because we are trying to do multiclass classification. Cross entropy loss was used as a loss function for the output. Adam was used as the optimization method instead of Stochastic Gradient Descent for this neural net because Adam outperforms Stochastic Gradient Descent by adapting learning rates for each parameter based on the average first moment and also takes advantage of the second moment. Using this optimization method leads to a classification accuracy on the test set of 91.775% as shown below:

```
The hit rate for Neural Net is: 0.91775
```

b) SVM: q3_2_svm.py

The SVM in my implementation was tuned using a grid search tool on two different kernel types: Gaussian and Linear; and four distinct sizes of margins: 1, 10, 100, and 1000. The accuracies and best parameters are shown below:

```
Test score using precision as metric is: 0.9704500132997163
{'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}
```

c) AdaBoost: q3_2_ada.py

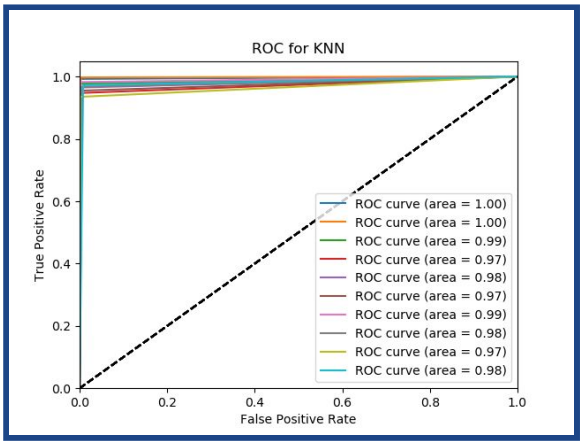
The AdaBoost in my implementation uses the decision tree of depth 1 as the weak classifier. The number of estimators was tuned on four different numbers: 10, 25, 50, and 100. The learning rate was also tuned, on five different numbers: 0.1, 0.5, 0.75, 1

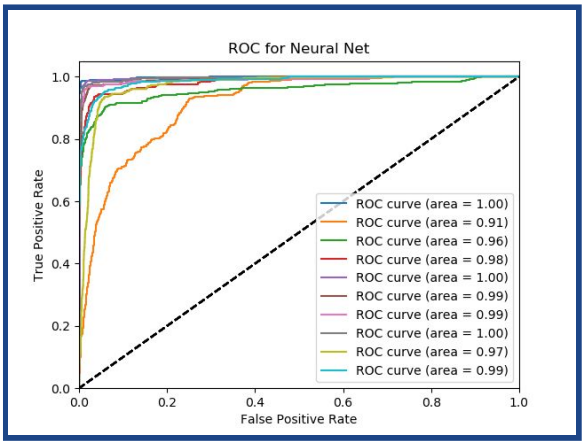
and 2. After using a grid search tool, the best parameters and the corresponding accuracy is as the following:

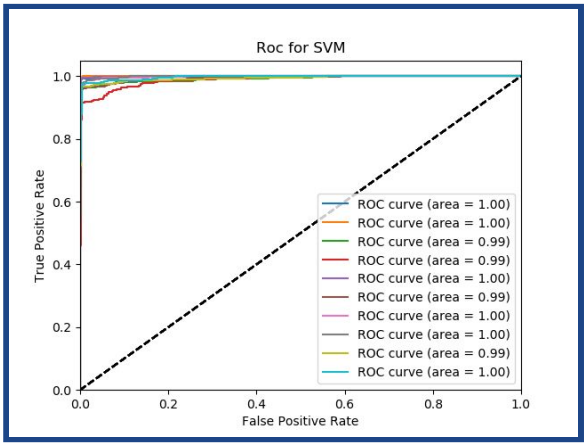
```
The best parameters are:
{'learning_rate': 0.5, 'n_estimators': 50}
The test score for AdaBoost is 0.804
```

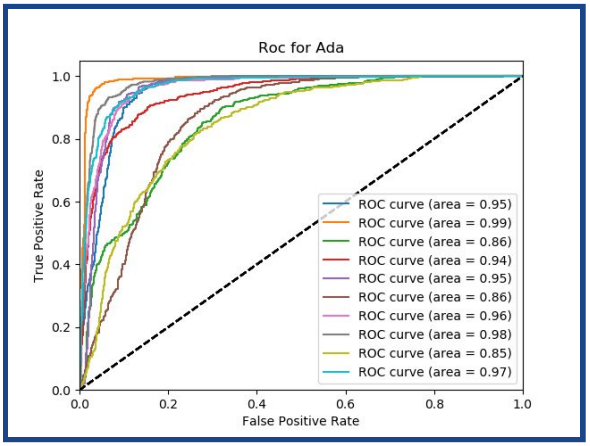
3) All the graphs from below can be generated by running the corresponding python files.

ROC Graphs:

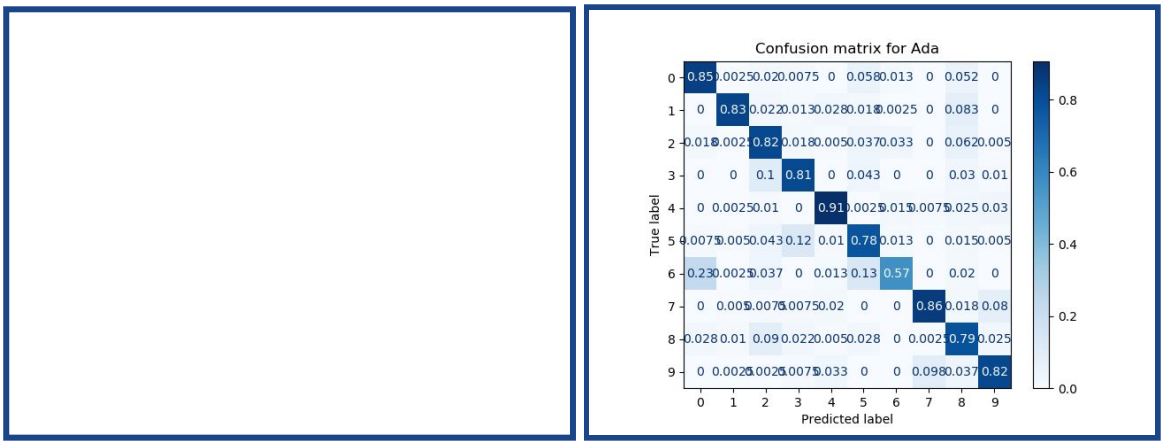


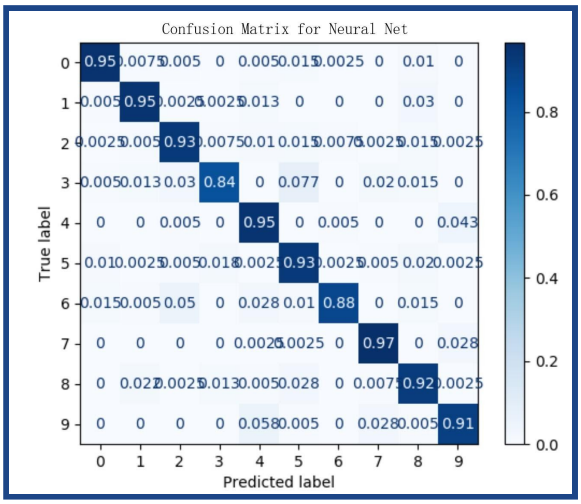


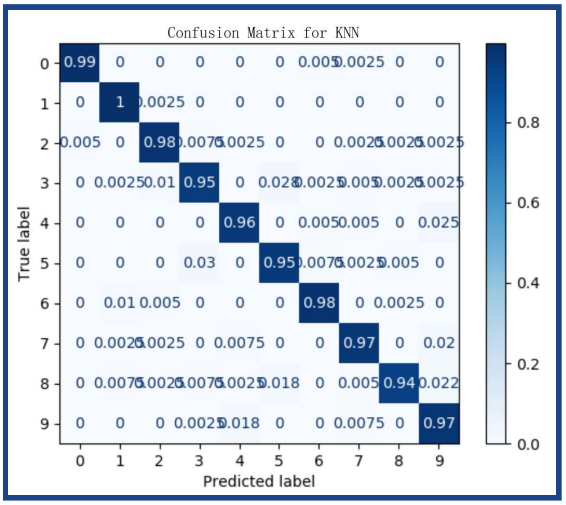




Confusion Matrices:







Accuracy, Precision and Recall:

```
The accuracy score for Ada is: 0.804
The precision score for Ada is: 0.8135884125796929
The recall score for Ada is: 0.8039999999999999
```

```
The accuracy score for KNN is: 0.96925
The precision score for KNN is: 0.9694943282680057
The recall score for KNN is: 0.9692500000000001
```

```
The accuracy score for Neural Net is: 0.9135
The precision score for Neural Net is: 0.9172144629341851
The recall score for Neural Net is: 0.9135
```

```
The accuracy score for SVM is: 0.9705
The precision score for SVM is: 0.9704500132997163
The recall score for SVM is: 0.9705
```

From the ROC graphs, we can see that the SVM has the greatest area underneath each curve and all of the areas are almost 1. This indicates that SVM has an accuracy of almost 1 in predicting each digit. However, the graph for AdaBoost has much flatter curves, which indicates a deficient performance.

The confusion matrices provide a general view of how good each classifier do in classifying each sample of a certain class. The diagonal of each confusion matrix shows the percentage for a classifier to classify a sample to the right class. From the graph, we can clearly see that the confusion matrix for SVM has the greatest average number along the diagonals, while the numbers along the diagonals on the graph for AdaBoost are not so well.

Accuracy, precision and recall are all real numbers calculated on the entire dataset, so we can directly compare these numbers across all classifiers and see that SVM has the best performance and AdaBoost performs the worst.

My expectation is that SVM will perform is best in general and AdaBoost will be the worst classifier, and the result matches my expectation by comparing different error metrics. This

result is reasonable because images have low intrinsic dimensions. This means similar images usually lie close to each other in high dimensional spaces while two non-similar images can be very far away from each other. This suggests that decision boundaries can be easily computed and will give a good performance in practice, so SVM does so well in classifying images. AdaBoost performs the worst because the weak learner being used are decision trees and decision trees classify images using only one pixel of the image, which will not provide a sufficient amount of information. In other words, the weak learner was too weak and even a huge collection of them cannot do well.