1. a) The equation $z_1 = W^{(1)}x$ with $x \in \mathbb{R}^d$ implies that $W^{(1)}$ has $d$ columns and $k \geq 1$ rows so $W^{(1)}$ has dimensions $k \times d$ for some $k \geq 1$. Thus, $z_1$ is a column vector with $k$ elements so $z_1$ has dimensions $k \times 1$ for some $k \geq 1$.

The equation $z_2 = h + x$ with $x, h \in \mathbb{R}^d$ implies that $z_2 \in \mathbb{R}^d$ so $z_2$ has dimensions $d \times 1$.

The equation $\mathcal{L} = \frac{1}{2}(y-t)^2$ with $t \in \mathbb{R}$ implies $y \in \mathbb{R}$. The equation $y = W^{(2)}z_2$ where $y \in \mathbb{R}$ and $z_2$ has dimensions $d \times 1$ implies $W^{(2)}$ has dimensions $1 \times d$.

b) The parameters are all the elements in $W^{(1)}$ and $W^{(2)}$ as the features $x$, targets $t$, and activation function $\sigma$ are already known. $W^{(1)}$ has $kd$ elements for some integer $k \geq 1$ and $W^{(2)}$ has $(1)(d) = d$ elements so there are $kd + d = (k+1)d$ parameters in the network for some $k \geq 1$.

c) A derivative expression indicate that each element of the variable has its derivative taken separately, and all of the resulting derivatives are aggregated into a single vector. The results on slide 44 of lecture 5 are used.

$$\begin{aligned}
\bar{y} &= \frac{d}{dy}\left(\frac{1}{2}(y-t)^2\right) \\
&= \frac{1}{2}(2)(y-t)\frac{d}{dy}(y-t) \\
&= (y-t)(1) \\
&= y - t
\end{aligned}$$

$$\begin{aligned}
\bar{W}^{(2)} &= \bar{y}\frac{d}{dW^{(2)}}\left(W^{(2)}z_2\right) \\
&= (y-t)z_2^T \\
&= (W^{(2)}z_2 - t)z_2^\intercal
\end{aligned}$$

$$\begin{aligned}
\bar{z}_2 &= \bar{y}\frac{d}{dW^{(2)}}\left(W^{(2)}z_2\right) \\
&= (y-t)W^{(2)\intercal} \\
&= W^{(2)\intercal}(W^{(2)}(h+x) - t)
\end{aligned}$$

$$\begin{aligned}
\bar{h} &= \bar{z}_2\frac{d}{dh}(h+x) \\
&= (W^{(2)}z_2 - t)W^{(2)\intercal}(1) \\
&= (W^{(2)}(h+x) - t)W^{(2)\intercal}
\end{aligned}$$

$$\begin{aligned}
\bar{z}_1 &= \bar{z}_2\frac{d}{dz_1}\sigma(z_1) \\
&= (W^{(2)}(h+x) - t)W^{(2)\intercal} \odot \sigma'(z_1)
\end{aligned}$$

where $\odot$ is element-wise multiplication.

$$\begin{aligned}\bar{W^{(1)}} &= \bar{z_1}\frac{d}{dW^{(1)}}\left(W^{(1)}x\right) \\ &= [(W^{(2)}(h+x)-t)W^{(2)\intercal} \odot \sigma'(z_1)]x^\intercal\end{aligned}$$

$$\begin{aligned}\bar{x} &= \bar{z_1}\frac{d}{dx}\left(W^{(1)}x\right) \\ &= W^{(1)\intercal}[(W^{(2)}(h+x)-t)W^{(2)\intercal} \odot \sigma'(z_1)]\end{aligned}$$

2. a) If $k \neq k'$, then using the quotient rule gives

$$\begin{aligned}
\frac{\partial y_k}{\partial z_{k'}} &= \frac{\left(\frac{d}{dz_{k'}} \exp(z_k)\right)\left(\sum_{j=1}^K \exp(z_j)\right) - (\exp(z_k))\frac{d}{dz_{k'}}\left(\sum_{j=1}^K \exp(z_j)\right)}{\left(\sum_{j=1}^K \exp(z_k)\right)^2} \\
&= \frac{(0)\left(\sum_{j=1}^K \exp(z_j)\right) - (\exp(z_k))(\exp(z_k) + \sum_{j \neq k'} 0)}{\left(\sum_{j=1}^K \exp(z_k)\right)^2} \\
&= -\frac{\exp(z_k)\exp(z_{k'}}{\left(\sum_{j=1}^K \exp(z_k)\right)^2} \\
&= -y_k y_{k'}
\end{aligned}$$

If $k = k'$, the using the quotient rule gives

$$\begin{aligned}
\frac{\partial y_k}{\partial z_{k'}} &= \frac{\left(\frac{d}{dz_k} \exp(z_k)\right)\left(\sum_{j=1}^K \exp(z_j)\right) - (\exp(z_k))\frac{d}{dz_k}\left(\sum_{j=1}^K \exp(z_j)\right)}{\left(\sum_{j=1}^K \exp(z_k)\right)^2} \\
&= \frac{(\exp(z_k))\left(\sum_{j=1}^K \exp(z_j)\right) - (\exp(z_k))(\exp(z_k) + \sum_{j \neq k} 0)}{\left(\sum_{j=1}^K \exp(z_k)\right)^2} \\
&= \frac{\exp(z_k)}{\sum_{j=1}^K \exp(z_k)} - \frac{(\exp(z_k))^2}{\left(\sum_{j=1}^K \exp(z_k)\right)^2} \\
&= y_k - y_k^2
\end{aligned}$$

b) Suppose the $n$th element of $\mathbf{t}$ is 1 so

$$\begin{aligned}\mathcal{L}_{\text{CE}}(\mathbf{t}, \mathbf{y}) &= -\sum_{k=1}^{K} t_k \log y_k \\ &= (1) \log y_n + \sum_{k \neq n} (0) \log y_k \\ &= \log y_n\end{aligned}$$

Then

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{CE}}(\mathbf{t}, \mathbf{y})}{\partial \mathbf{w}_k} &= \frac{\partial}{\partial \mathbf{w}} \log y_n \\ &= \frac{1}{y_n} \frac{\partial y_n}{\partial \mathbf{w}_k} \\ &= \frac{1}{y_n} \frac{\partial y_n}{\partial \mathbf{w}_k}\end{aligned}$$

Let $h(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \mathbf{w}_k \mathbf{x} = z_k$ so $y_n = \text{softmax}(h(\mathbf{w}_1, \ldots, \mathbf{w}_K))_k$ where the $z_i$ for $i \neq k$ are constant in $\text{softmax}_k$. Thus,

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{CE}}(\mathbf{t}, \mathbf{y})}{\partial w_i} &= \frac{1}{y_n} \frac{\partial}{\partial \mathbf{w}_k} \text{softmax}(h(\mathbf{w}_1, \ldots, \mathbf{w}_K))_k \\ &= \frac{1}{y_n} \frac{\partial y_n}{\partial h} \frac{\partial h}{\partial \mathbf{w}_k} \qquad (1)\end{aligned}$$

Using a), we have $\frac{\partial y_n}{\partial h} = \frac{\partial y_n}{\partial \mathbf{z}}$ has $i$th element equal to $y_n y_i$ if $i \neq n$ and $y_n - y_n^2$ if $i = n$. Then $\frac{1}{y_n} \frac{\partial y_n}{\partial z_i}$ has $i$th element equal to $\frac{-y_n y_i}{y_n} = 0 - y_i = t_i - y_i$ for $i \neq n$ and $\frac{y_n - y_n^2}{y_n} = 1 - y_n = t_i - y_i$ if $i = n$. Thus, $\frac{1}{y_n} \frac{\partial y_n}{\partial z_i} = t_i - y_i$ for all $1 \leq i \leq K$ so

$$\frac{\partial y_n}{\partial h} = \frac{\partial y_n}{\partial z_k} = t_k - y_k \qquad (2)$$

Also, $\frac{\partial h}{\partial \mathbf{w}_k} = \frac{\partial}{\partial \mathbf{w}_k} \mathbf{w}_k \mathbf{x} = \mathbf{x}$. Substituting this and (2) into (1) gives

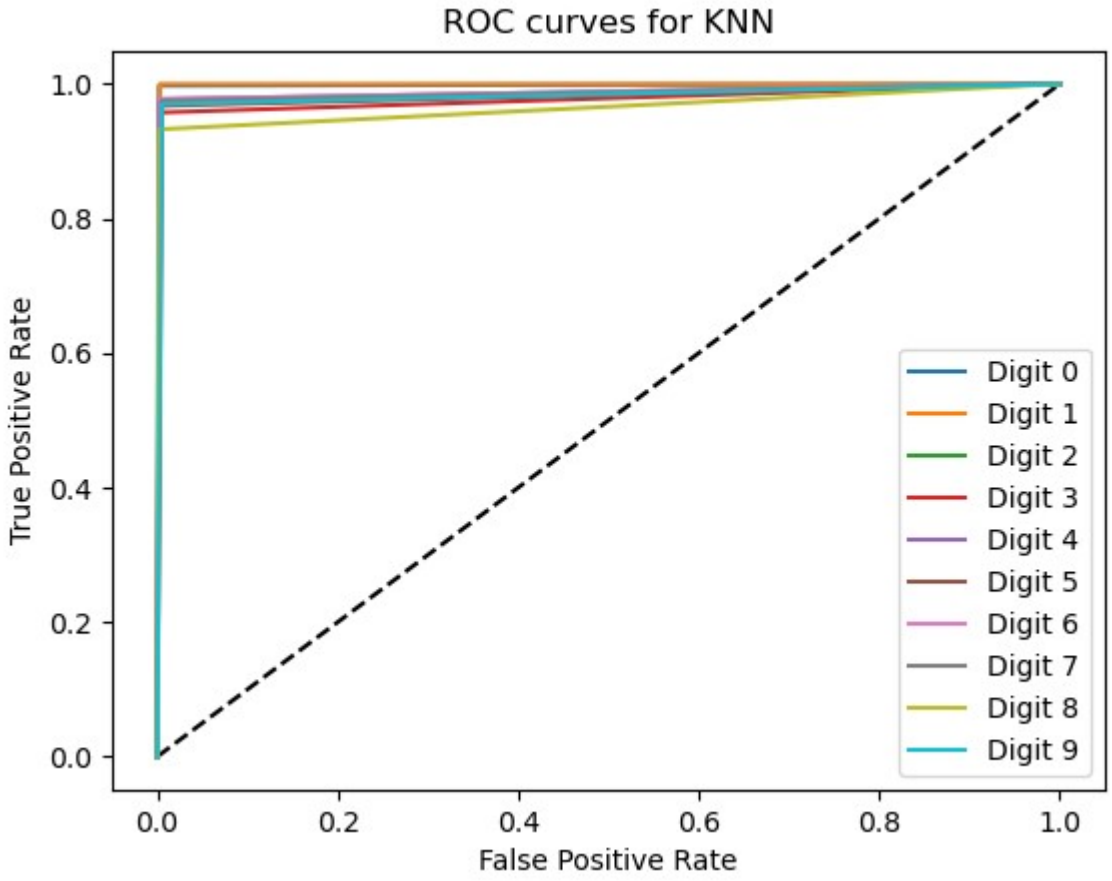
$$\frac{\partial \mathcal{L}_{\text{CE}}(\mathbf{t}, \mathbf{y})}{\partial w_i} = (t_k - y_k) \cdot \mathbf{x}$$

3.1.2 Ties are broken by whichever label has its first occurrence in the labels list appear before the first occurrence of the other tied labels in the labels list.

```
Accuracy of KNN classifier using 1 neighbour on train set is 1.0
Accuracy of KNN classifier using 1 neighbour on test set is 0.9688
Accuracy of KNN classifier using 15 neighbour on train set is 0.9637
Accuracy of KNN classifier using 15 neighbour on test set is 0.961
Mean accuracy of KNN classifier using 1 neighbours across 10 folds: 0.9644
Mean accuracy of KNN classifier using 2 neighbours across 10 folds: 0.9644
Mean accuracy of KNN classifier using 3 neighbours across 10 folds: 0.9651
Mean accuracy of KNN classifier using 4 neighbours across 10 folds: 0.9656
Mean accuracy of KNN classifier using 5 neighbours across 10 folds: 0.9634
Mean accuracy of KNN classifier using 6 neighbours across 10 folds: 0.9646
Mean accuracy of KNN classifier using 7 neighbours across 10 folds: 0.9607
Mean accuracy of KNN classifier using 8 neighbours across 10 folds: 0.9616
Mean accuracy of KNN classifier using 9 neighbours across 10 folds: 0.958
Mean accuracy of KNN classifier using 10 neighbours across 10 folds: 0.9569
Mean accuracy of KNN classifier using 11 neighbours across 10 folds: 0.9556
Mean accuracy of KNN classifier using 12 neighbours across 10 folds: 0.955
Mean accuracy of KNN classifier using 13 neighbours across 10 folds: 0.9531
Mean accuracy of KNN classifier using 14 neighbours across 10 folds: 0.9543
Mean accuracy of KNN classifier using 15 neighbours across 10 folds: 0.9497
The optimal KNN classifier uses 4 neighbours.
Accuracy of optimal KNN classifier (4 neighbours) on train set is 0.9864
Accuracy of optimal KNN classifier (4 neighbours) on test set is 0.9728
```

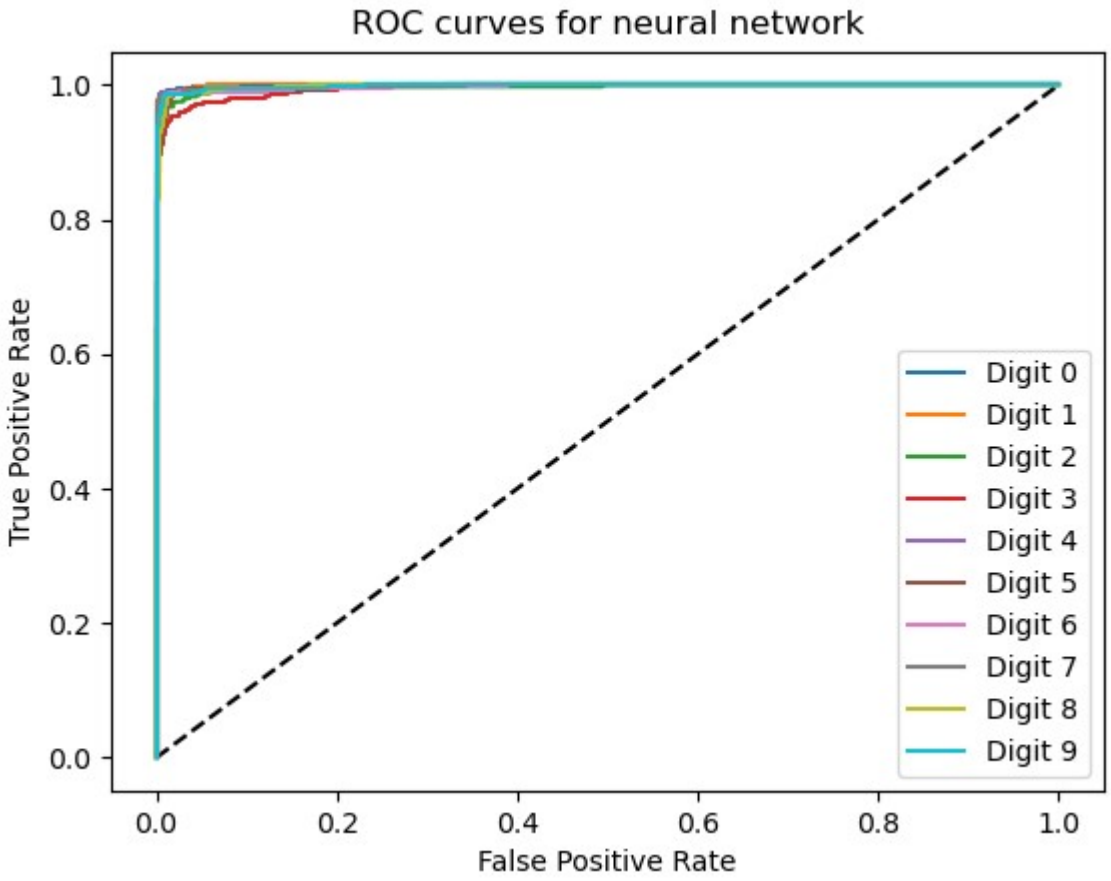
3.3

# KNN



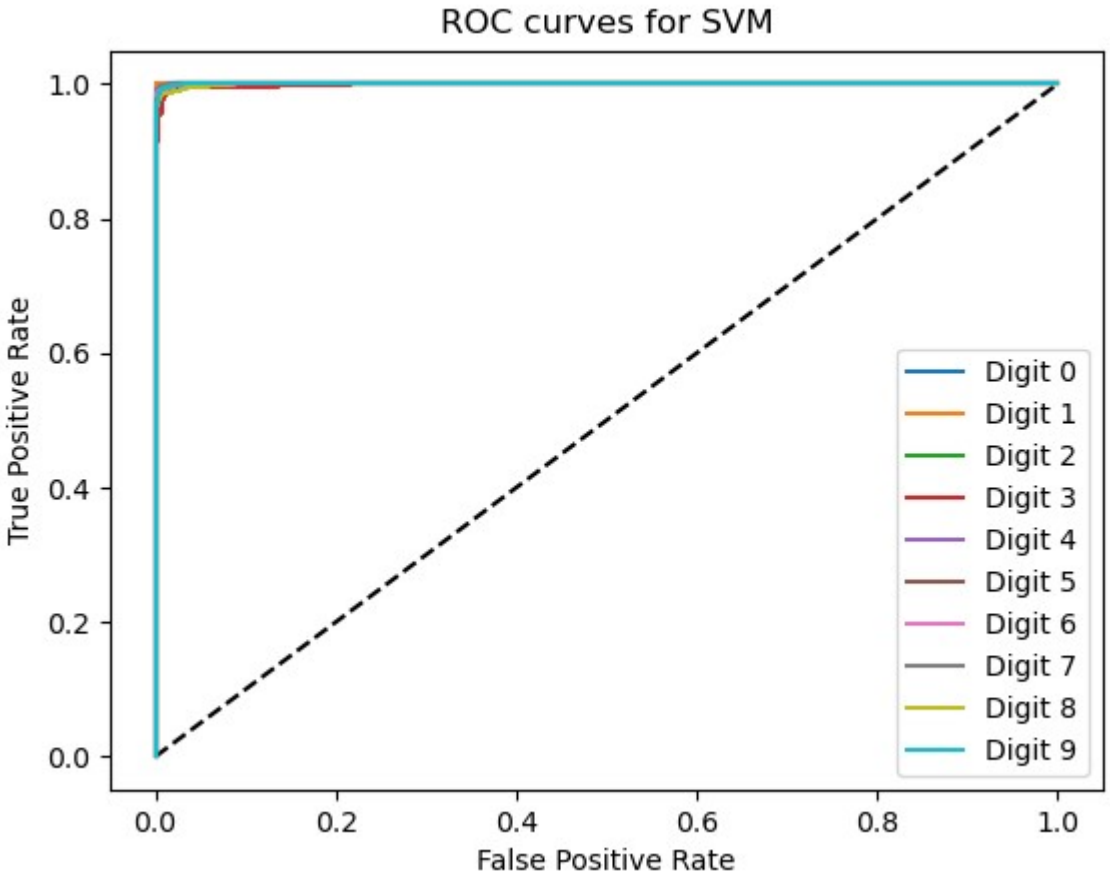

```
The accuracy of KNN is 0.973
The precision for each digit of KNN is [0.983 0.973 0.987 0.977 0.98  0.951 0.982 0.961 0.979 0.956]
The recall for each digit of KNN is [0.998 1.    0.972 0.958 0.978 0.968 0.978 0.975 0.932 0.97 ]
The confusion matrix of KNN is
 [[399   0   0   0   0   0   0   1   0   0]
 [  0 400   0   0   0   0   0   0   0   0]
 [  4   0 389   0   1   1   1   2   1   1]
 [  0   1   2 383   0   8   1   2   2   1]
 [  0   1   0   0 391   0   1   1   0   6]
 [  1   0   0   5   0 387   3   1   3   0]
 [  1   4   1   0   0   1 391   0   2   0]
 [  0   2   1   0   2   0   0 390   0   5]
 [  1   2   1   3   1  10   1   3 373   5]
 [  0   1   0   1   4   0   0   6   0 388]]
```

Neural network



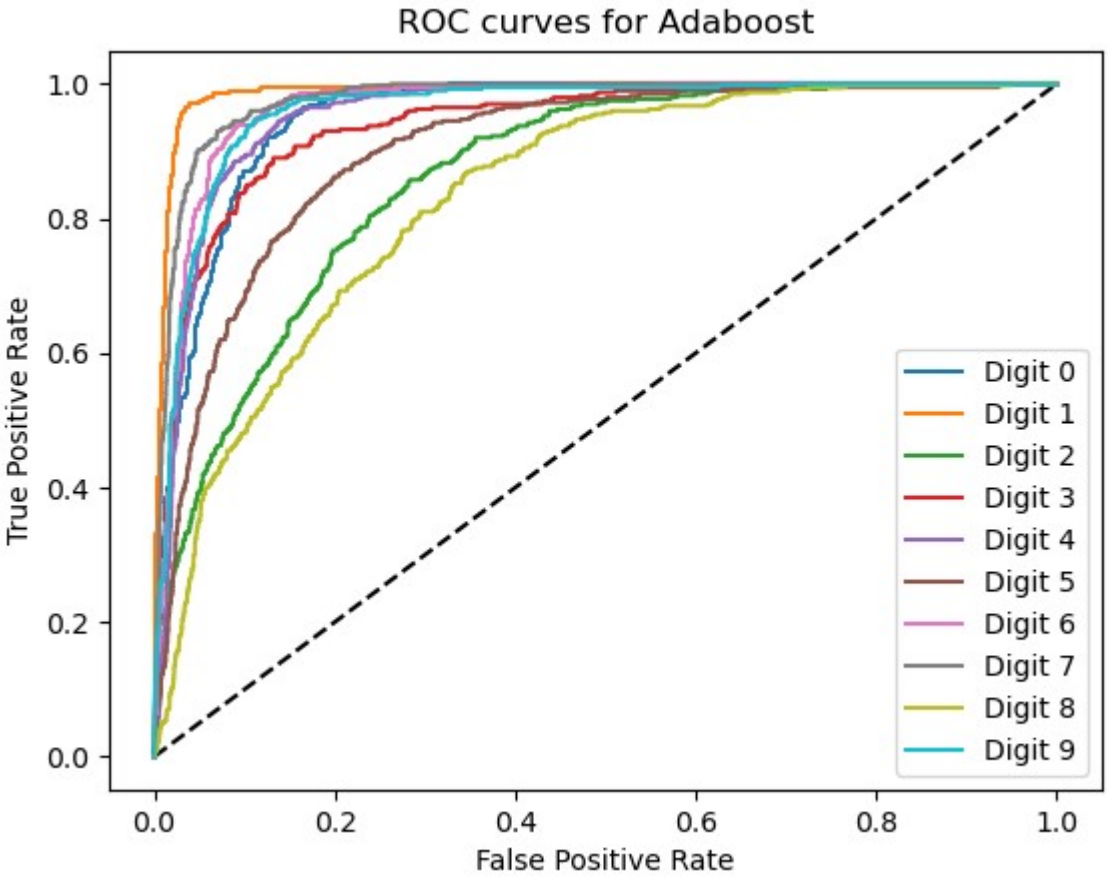

```
The accuracy of neural network is 0.956
The precision for each digit of neural network is [0.963 0.958 0.966 0.98  0.963 0.918 0.951 0.973 0.927 0.967]
The recall for each digit of neural network is [0.985 0.98  0.935 0.878 0.985 0.955 0.972 0.975 0.948 0.95 ]
The confusion matrix of neural network is
 [[394   3   0   0   2   0   0   0   1   0]
 [  0 392   0   1   2   0   2   0   3   0]
 [  1   1 374   4   1   2   9   1   6   1]
 [  5   1   8 351   0  23   1   2   7   2]
 [  0   0   0   0 394   0   4   0   0   2]
 [  3   2   0   1   0 382   4   2   6   0]
 [  2   3   2   0   3   1 389   0   0   0]
 [  0   0   1   0   1   0   0 390   1   7]
 [  3   6   1   0   1   7   0   2 379   1]
 [  1   1   1   1   5   1   0   4   6 380]]
```

SVM




```
The accuracy of SVM is 0.979
The precision for each digit of SVM is [0.993 0.99  0.982 0.964 0.971 0.968 0.982 0.98  0.977 0.98 ]
The recall for each digit of SVM is [0.998 0.998 0.978 0.948 0.992 0.978 0.982 0.98  0.962 0.972]
The confusion matrix of SVM is
 [[399   1   0   0   0   0   0   0   0   0]
 [  0 399   0   0   0   0   1   0   0   0]
 [  0   0 391   3   0   1   3   0   2   0]
 [  0   1   5 379   0   6   0   3   5   1]
 [  0   0   0   0 397   0   2   0   0   1]
 [  1   0   0   6   0 391   1   1   0   0]
 [  1   0   2   0   4   0 393   0   0   0]
 [  0   0   0   0   4   0   0 392   1   3]
 [  1   1   0   4   0   6   0   0 385   3]
 [  0   1   0   1   4   0   0   4   1 389]]
```

# Adaboost




```
The accuracy of Adaboost is 0.767
The precision for each digit of Adaboost is [0.703 0.944 0.715 0.711 0.791 0.738 0.83  0.877 0.696 0.756]
The recall for each digit of Adaboost is [0.782 0.838 0.735 0.825 0.832 0.745 0.562 0.732 0.822 0.798]
The confusion matrix of Adaboost is
 [[313   6  11   8   0  15   9   1  37   0]
 [  0 335  15   7  28   3   0   0  12   0]
 [ 17   1 294  12   9  16  15   1  34   1]
 [  0   0  28 330   0  25   0   0  14   3]
 [  0   4   2   1 333   0  19   2   6  33]
 [  3   2  10  62   6 298   2   1  14   2]
 [109   1  19   0   9  28 225   0   9   0]
 [  0   3   8  23   7   1   0 293   7  58]
 [  3   3  21  14   5  18   1   0 329   6]
 [  0   0   3   7  24   0   0  36  11 319]]
```

KNN, neural network, and SVM all have high accuracy, precision, and recall with SVM being slightly better than KNN, which in turn is slightly better than neural network. Adaboost performed the worst on all the metrics with accuracy, precision, and recall significantly less than the other models. For all the models, the confusion matrix has the largest values on the diagonal, which indicates most of the data is categorized correctly. There are entries not on the diagonal that also had large values, which correspond to mistaking a number for a number that looks similar such as 6 for 0 and 3 for 8.

KNN tends to be when the features are continuous or when they interact in complicated ways, which are both properties of the pixel value arrays. The values are also in $[0, 1]$ rather than than being large in magnitude and there were also no missing values, factors which could hinder the effectiveness of KNN.

In the neural network, ReLU was used, which does not frequently outputs 0 as the value unlike other activations such as the sigmoid. This cases the features to be less dependent, which is beneficial since there are many pixels in the images. The softmax layer creates a probability distribution that is useful for categories such as digits. Neural networks also benefit from a large training, which is the case with this set as there are an average of 700 examples for each digit.

SVM worked well because the kernel trick can be used with different kernel functions, which makes it possible to choose the optimal function for this digit recognition task. SVM also works well in high dimensional data such as the 64-dimensional pixel vectors. SVM also uses a regularization parameter, which prevented it from overfitting from the large amount of noise in the digits data.

Adaboost is sensitive to the number amount of noise in the data, which caused it not learn distinctions between the digits such as the most common mistake of predicting 0 in images of 6. Adaboost is also sensitive to outliers, which can more often occur due to the similar shape of many pairs of digits. Adaboost is good at generalization, which is not ideal for the digits task because one digit can easily be generalized to another digit.