Project 3 Report

Yiqi Chen

Department of Chemical and Biological Enginerring
University at Buffalo
Buffalo, NY 14228
yiqichen@buffalo.edu

Abstract

In this project, machine learning techniques were used to identify handwritten digits images. The Modified National Institute of Standards and Technology (MNIST) database was used to train and test five different types of machine learning models. The USPS handwritten digits dataset was also used to evaluate those models. The machine learning models investigated in this projects are: logistic regression, linear support vectors machines, random forest, deep neural networks (DNN) and covolutional neural networks (CNN). The optimum hyperparameters for those models were chosen by a grid search approach. At last, these models were combined by both a direct hard voting and boosting hard voting methods to make predictions. It is observed from the performances that all the models achieved an accuracy of more than 90% in the MNIST datasets and the CNN model is the most accurate one. However, for the USPS dataset, the highest accuracy these models could get is less than 60%. In addition, the combined models failed to provide an increase in performance from the effective CNN model.

1 Introduction

Classification and identification of handwritten digits are among the first important applications for machine learning techniques. The MNIST database (Modified National Institute of Standards and Technology database) is a famous database of handwritten digits which is commonly used in training and evaluating machine learning models. In this project, five different types of machine learning models (logistic regression, linear support vector machine, random forest, deep neural networks and convolutional neural networks) were trained on the MNIST data. A grid search for hyperparameters was applied to all the models to get the best performances af these models. They were then evaluated in the MNIST test dataset as well as a data set from the USPS database. At last, all five models were combined by both a direct hard voting and boosting hard voting approaches as an effort to achieve a better performance. The objective of this project is to explore the strength and weakness of different machine learning models.

2 Methods and Models

2.1 Data Reading and Partition

The MNIST dataset was read in and partitioned into training, validation and testing sets. 5000 images were included in the training set and 1000 images were in both validation and testing sets. Since all those images are in gray scale (28×28), the dimension of the data is 784. A USPS handwritten digits database was also used for testing how well the models in this project generalize. A total of 20000 segmented and cropped images at a resolution of 100ppi are in the USPS database. They were

Preprint. Work in progress.

resized to 28×28 like the MNIST digits and then stored as a new testing set. The one-hot encoding process was applied to transfer the target vector to a sparse matrix ($N_{samples} \times K_{classes}$) where each column corresponds to one possible value of targets. This process can remove the effect of natural ordering between targets and improve the model performance.

2.2 Logistic Regression Model

The first model used in this project is the logistic regression model. Logistic regression is a regression analysis that works very well with categorical targets. It usually link target y to weight w and input x by an activation function σ:

$$y(x, w) = \sigma(w^T X) \tag{1}$$

When dealing with a multi-class classification problem, the softmax function is usually chosen to be the activation function. Let $a = w^T X$, the target for class i is expressed as

$$y_i = \frac{exp(a_i)}{\sum_j^K exp(a_j)} \tag{2}$$

where K is the number of categories. The softmax function introduces nonlinearity into the input-output relationship and its output is effectively the normalized probability of each target class. So the likelihood function of logistic regression is

$$p(t|W) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|x_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}} \tag{3}$$

where t is the target matrix, N is the total number of samples and K the number of categories as stated. The error function in this model, which is also known as cross-entropy error, is derived by taking the negative logarithm of the likelihood function:

$$E(W) = -\ln p(t|W) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk} \tag{4}$$

Finding the value of W that minimizes the error function is equivalent to setting the derivative of error function to zero ($\nabla E(w) = 0$). In stochastic gradient descent algorithm, the equation is solved by an iterative approach which begins with an initial guess of the weight matrix and then the updates it by $W^{\tau+1} = W^{\tau} - \eta \nabla E_n$. Here $\nabla E_n = (y_n - t_n)x_n$ is the gradient of error for one sample of data. The two hyperparameters tuned for this model were the learning rate and the regularization factor.

2.3 Support Vector Machines

The Support Vector Machines (SVM) is the second model tested in this project. SVM is a discriminative method which relies on a linear combination of a kernel evaluated at the training samples to make predictions. The key idea of SVM, which is also called maximum margin classifiers, is to find the best decision boundary(ies) by maximizing margin. Margin is defined as the smallest distance between decision boundary and any of the samples. So data points from different classes are well separated when margin is at maximum. In a binary classifier, because we want $t_n y(x_n) > 0$ for all n, the maximum margin solution is found by

$$\underset{w,b}{\text{argmax}}\{\frac{1}{|w|} \min_n [t_n(w^T \phi(x_n) + b)]\} \tag{5}$$

where w is weight vector, t is target vector, b is the bias term and $\phi(x_n)$ the basis function of training data x. Solving this equation is equivalent to minimizing a Lagrange function L with respect to w and b.

$$L(w, b, a) = \frac{1}{2}|w|^2 - \sum_{n=1}^{N} a_n[t_n(w^t \phi(x_n) + b) - 1] \tag{6}$$

where a_n is a positive multiplier. After eliminating w and b in this function, the problem becomes maximizing

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_m a_n t_n t_m k(x_n, x_m) \tag{7}$$

subject to $\sum_{n=1}^{N} a_n t_n = 0$. Here $k(x, x^{'})$ is the kernel function which can effectively reduce computational cost in high dimensional space. There are many possible choices of kernel functions. In this project, a linear kernel and a Gaussian kernel were tested and evaluated.

2.4 Random Forest

The third model tested was the random forest classifier. It is an ensemble learning method used for tackling both classification and regression problems. It randomly constructs a "forest" of decision trees during training process and makes predictions by taking a majority vote (classification) or mean value (regression) of individual trees. A decision tree classifier uses a tree-like structure to make conclusion by traversing from the input data at root to the class labels at leaf nodes. In such a tree, each node specifies a feature and each branch descending from a node represents a possible value of this feature. In this setup, a decision needs to be made every time when moving down to the next layer of the tree. One big disadvantage of the decision trees method is it constantly overfits the training dataset. Random forest algorithm is developed to fix this issue. By randomly selecting a subset of both data points and features from the training set in a bootstrap fashion, random forest model generalizes much better than decision trees method. In this project, random forest classifier was optimized by choosing the best number of trees in the forest.

2.5 Neural Networks Model

The last two models used in this project were in the neural networks family. The neural networks are a biologically-inspired machine learning models consisted with interconnected units and nodes. Two types of neural network models were tested in this project: deep neural network and covolutional neural network.

Deep Neural Networks Deep neural networks (DNN) model is an artificial neural network which has multiple layers between input and output data. In this project, a two layer neural network model was trained to predict the output. The activation function in every hidden layer is the "relu" function. For the final output layer, the "softmax" function is used. The model was compiled by the "Adam" optimizer based on the categorical cross entropy error (eq 4). Two hyperparameters were tuned in the model: number of nodes in the hidden layers and dropout rate.

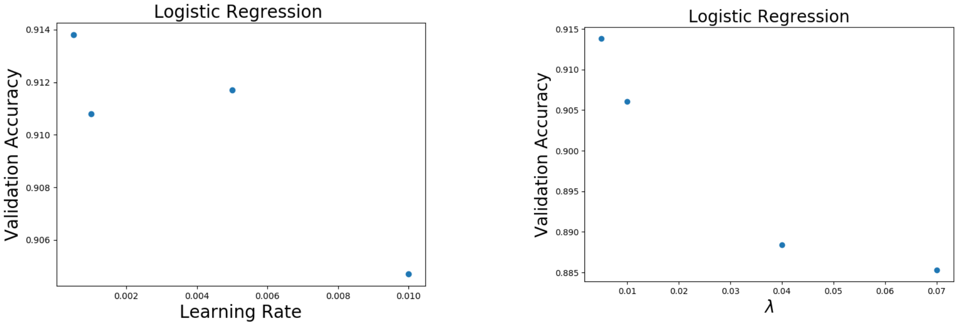
Convolutional Neural Networks Convolutional neural networks (CNN) model is a unique class of deep neural networks which is best suited to deal with machine learning problems containing image analysis. Compared to traditional neural networks, CNN uses multiple convolutional layers to process training data before feeding it to the fully connected neural networks. Such an architecture dramatically reduces computation costs while still preserves the essence of data. To performance a convolution process, the input image first needs to converted into a pixel matrix (28×28 in this project). Then a convolution kernel is applied to the matrix to convolve the image. Next a pooling function is operated on the new matrix to extract the most meaning feature. Here I choose a max pooling function which returns the maximum output within a rectangular neighborhood. The pooling operation reduces the dimensionality of features and reduces computational costs. In addition, it helps prevent overfitting. After going through multiple convolutional layers, the input data is converted back to a vector and fed into a fully connected DNN to produce outputs. In this project, considering the large size of training data, only one covolutional layer and one fully connected layer was constructed. The two parameters tuned for CNN were the number of filters and dropout rates.

2.6 Combining Models

In the last part of this project, all the models used were combined into a more powerful classifier to provide more precise predictions. The simplest method for combining them is the "hard voting" approach in which a majority voting among all classifiers is taken for every data point. However, simply using this method can not guarantee improvement in model performance since all models are trained individually without any correlation. A better approach is the boosting method where different models are trained in sequential. This method has two steps. In the first step, every model trains on a subset of the original training data set. The creation of subsets is not totally random. It depends upon the performance of previous models: the data points that were misclassified by previous models have higher weights in the selection of next training. This setup emphasizes the improvements on

Table 1: Performance of the Optimum Logistic Regression Model

DataSets	RMS Error	Accuracy
MNIST_Training	0.8546489023045206	0.90598
MNIST_Validation	0.8233056318725186	0.9148
MNIST_Test	0.8341381940823845	0.9125
USPS_Test	2.16991883186397	0.3648182409120456



(a) Accuracy with Different Learning Rates

(b) Accuracy with Different Regularization Factors

Figure 1: Logistic Regression Performances with Different Parameters

the weakness in individual models and usually produces better performances. In the second step, predictions from these models are combined by the "hard voting" system.

3 Results and Discussions

3.1 Logistic Regression

A stochastic gradient descent logistic regression model was constructed in this project. During the training process, the input data was fed to the model repeatedly for 20 times. A grid search was carried out to find the optimum hyperparameters. Every combination of learning rate lr in $\{0.0005, 0.001, 0.005, 0.01\}$ and regularization factor λ in $\{0.005, 0.01, 0.04, 0.07\}$ was tested and evaluated by the accuracy in the validation set. The performances with different parameters are shown in fig 1. The parameters that generated the highest accuracies are: $lr = 0.001$ and $\lambda = 0.005$. Then the optimum model was created and evaluated. Table 1 shows the performance on all three MNIST sets as well as the USPS test set. Fig 2 and fig 3 are the confusion matrices for MNIST and USPS test sets, respectively. Those results show that the model is able to correctly predict more than 90% of the samples in MNIST training, validation and test sets. However, only $\sim 36\%$ of the USPS data points were successfully identified. The difference between these prediction accuracies validate the “No Free Lunch” theorem which claims that there is no one model that works best for every problem. Models trained by MNIST datasets can not generalize very well in other datasets.

3.2 Support Vector Machines

In this section, results from different kinds of support vector machines are reported. The skit-learn package was used in construct the models. SVC models with both linear and Gaussian kernels were tested and evaluated. For Gaussian kernels, different setup of parameter γ were used: $\gamma = 1.0$ and $\gamma =' auto'$. γ is the inverse of the radius of influence of samples selected by the model as support vectors. The behavior of the SVC model is very sensitive to the γ parameter. If it is too large, the radius of the area of influence of the support vectors is very small and the model is not able to produce good performance. The "auto" method sets $\gamma = 1/N_{samples}$ and this is usually a good value to use. Because the extreme long time required to train a Gaussion SVC model, the tuning of the regularization parameter C was only done for the linear SVC. Four possible values, $\{1.0, 2.0, 5.0, 10.0\}$, were tested. Fig 4 shows the validation set accuracy for these different values

$$\begin{bmatrix}
965 & 0 & 3 & 2 & 0 & 1 & 5 & 1 & 3 & 0 \\
0 & 1106 & 2 & 4 & 1 & 2 & 4 & 1 & 15 & 0 \\
11 & 7 & 902 & 16 & 15 & 1 & 13 & 16 & 45 & 6 \\
4 & 0 & 21 & 910 & 0 & 31 & 3 & 11 & 20 & 10 \\
2 & 6 & 6 & 1 & 910 & 0 & 8 & 2 & 8 & 39 \\
11 & 5 & 4 & 38 & 9 & 758 & 13 & 10 & 35 & 9 \\
16 & 3 & 4 & 2 & 11 & 17 & 900 & 1 & 4 & 0 \\
4 & 16 & 24 & 7 & 7 & 0 & 0 & 936 & 3 & 31 \\
8 & 9 & 8 & 27 & 8 & 31 & 12 & 14 & 846 & 11 \\
11 & 8 & 4 & 12 & 38 & 11 & 0 & 26 & 7 & 892
\end{bmatrix}$$

Figure 2: Confusion Matrix by Logistic Regression Model on MNIST Test Set

$$\begin{bmatrix}
649 & 2 & 255 & 56 & 227 & 129 & 66 & 51 & 164 & 401 \\
180 & 405 & 15 & 315 & 310 & 66 & 35 & 355 & 297 & 22 \\
200 & 23 & 1203 & 164 & 67 & 79 & 80 & 76 & 81 & 26 \\
90 & 3 & 140 & 1281 & 14 & 278 & 9 & 71 & 78 & 36 \\
61 & 82 & 39 & 52 & 1005 & 82 & 29 & 173 & 307 & 170 \\
172 & 24 & 129 & 180 & 43 & 1152 & 97 & 78 & 92 & 33 \\
310 & 14 & 392 & 103 & 113 & 291 & 695 & 10 & 50 & 22 \\
188 & 197 & 255 & 493 & 61 & 93 & 21 & 332 & 307 & 53 \\
249 & 32 & 143 & 217 & 114 & 632 & 96 & 53 & 398 & 66 \\
41 & 167 & 131 & 450 & 140 & 77 & 13 & 431 & 374 & 176
\end{bmatrix}$$

Figure 3: Confusion Matrix by Logistic Regression Model on USPS Test Set

of C. It's clear that the value of 1.0 is the best choice. Tables 2 through 4 present the comparison of the three different SVC models with $C = 1.0$. Figs 5 through 10 are the confusion matrices for linear kernel, Gaussian kernel with $\gamma = 1.0$ and $\gamma = auto$ on both MNIST test set and USPS test set, respectively. It can be seen that the linear SVC model achieved an accuracy of ~ 0.94 on the MNIST test set, which is an improvement compared to logistic regression model. However, the accuracy on the USPS test set is still very low. The "No Free Lunch" theorem is still effective. For the Gaussian kernels, the auto γ model produced a similar accuracy with the linear model. However, the one with $\gamma = 1.0$ seemed not converged at all as it is totally unable to make viable classifications on the test sets. The reason for that is the γ value is set too large.

3.3 Random Forest

In this suction, the performance from the random forest classifier is reported. The skit-learn package was used to construct the model. Six different number of trees, $\{10, 25, 50, 100, 150, 200\}$, were evaluated. The result is shown in fig 11. A random forest model containing 200 decision trees produced the highest accuracy on the validation set. Table 5 shows the accuracy on the three MNIST sets as well as the USPS test set. Fig 12 and fig 13 are the confusion matrices on the MNIST test and the USPS test set, respectively. The accuracies on the MNIST validation and test sets were about ~ 0.97. It is 3% higher than the SVC model performance. For the USPS test set, a improvement is

Table 2: Performance of Linear SVC Model

DataSets	Accuracy
MNIST_Training	0.97246
MNIST_Validation	0.9423
MNIST_Test	0.939
USPS_Test	0.3272163608180409

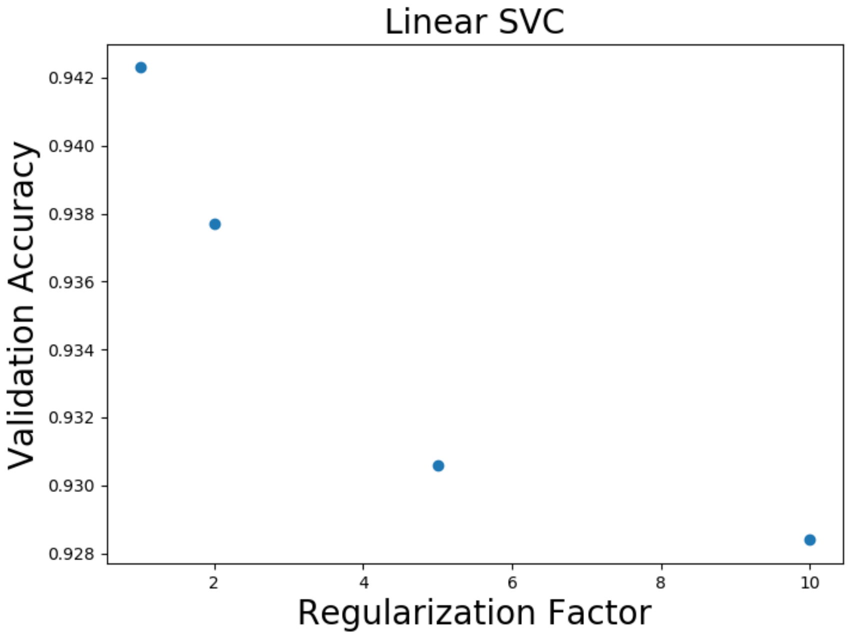


Figure 4: Linear SVC Model Performances with Different C

Table 3: Performance of Gaussian SVC with (gamma=1.0) Model

DataSets	Accuracy
MNIST_Training	1.0
MNIST_Validation	0.1824
MNIST_Test	0.1759
USPS_Test	0.10000500025001251

Table 4: Performance of Gaussian SVC with (gamma=auto) Model

DataSets	Accuracy
MNIST_Training	1.0
MNIST_Validation	0.1824
MNIST_Test	0.1759
USPS_Test	0.10000500025001251

$$\begin{bmatrix}
959 & 0 & 5 & 2 & 2 & 4 & 7 & 0 & 1 & 0 \\
0 & 1121 & 3 & 3 & 0 & 1 & 2 & 1 & 4 & 0 \\
6 & 8 & 968 & 9 & 3 & 2 & 11 & 10 & 13 & 2 \\
5 & 2 & 17 & 944 & 4 & 13 & 1 & 8 & 13 & 3 \\
2 & 1 & 10 & 1 & 943 & 0 & 4 & 2 & 2 & 17 \\
13 & 4 & 2 & 39 & 5 & 792 & 9 & 1 & 22 & 5 \\
10 & 3 & 11 & 1 & 5 & 14 & 911 & 2 & 1 & 0 \\
1 & 8 & 20 & 10 & 6 & 1 & 0 & 961 & 3 & 18 \\
8 & 4 & 9 & 25 & 11 & 27 & 6 & 5 & 871 & 8 \\
7 & 6 & 2 & 13 & 32 & 4 & 0 & 18 & 7 & 920
\end{bmatrix}$$

Figure 5: Confusion Matrix by Linear SVC Model on MNIST Test Set

$$\begin{bmatrix}
483 & 3 & 320 & 70 & 254 & 298 & 58 & 125 & 14 & 375 \\
60 & 475 & 136 & 300 & 342 & 167 & 22 & 413 & 62 & 23 \\
176 & 91 & 1163 & 126 & 55 & 209 & 60 & 62 & 37 & 20 \\
70 & 59 & 294 & 922 & 14 & 509 & 5 & 54 & 54 & 19 \\
26 & 25 & 136 & 76 & 884 & 194 & 9 & 463 & 84 & 103 \\
60 & 17 & 166 & 250 & 85 & 1203 & 36 & 50 & 104 & 29 \\
168 & 23 & 730 & 50 & 154 & 312 & 539 & 16 & 4 & 4 \\
26 & 85 & 190 & 664 & 60 & 310 & 13 & 526 & 89 & 37 \\
104 & 21 & 272 & 391 & 136 & 736 & 78 & 58 & 185 & 19 \\
15 & 50 & 163 & 513 & 165 & 108 & 7 & 623 & 192 & 164
\end{bmatrix}$$

Figure 6: Confusion Matrix by Linear SVC Model on USPS Test Set

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 980 & 0 & 0 \\
0 & 731 & 0 & 0 & 0 & 0 & 0 & 404 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1032 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1010 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 982 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 892 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 958 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1028 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 974 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1009 & 0 & 0
\end{bmatrix}$$

Figure 7: Confusion Matrix by Gaussian SVC (gamma=1.0) Model on MNIST Test Set

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1999 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2000 & 0 & 0
\end{bmatrix}$$

Figure 8: Confusion Matrix by Gaussian SVC (gamma=1.0) Model on USPS Test Set

$$\begin{bmatrix}
967 & 0 & 1 & 0 & 0 & 5 & 4 & 1 & 2 & 0 \\
0 & 1120 & 2 & 3 & 0 & 1 & 3 & 1 & 5 & 0 \\
9 & 1 & 962 & 7 & 10 & 1 & 13 & 11 & 16 & 2 \\
1 & 1 & 14 & 950 & 1 & 17 & 1 & 10 & 11 & 4 \\
1 & 1 & 7 & 0 & 937 & 0 & 7 & 2 & 2 & 25 \\
7 & 4 & 5 & 33 & 7 & 808 & 11 & 2 & 10 & 5 \\
10 & 3 & 4 & 1 & 5 & 10 & 924 & 0 & 1 & 0 \\
2 & 13 & 22 & 5 & 7 & 1 & 0 & 954 & 4 & 20 \\
4 & 6 & 6 & 14 & 8 & 24 & 10 & 8 & 891 & 3 \\
10 & 6 & 0 & 12 & 33 & 5 & 1 & 14 & 6 & 922
\end{bmatrix}$$

Figure 9: Confusion Matrix by Gaussian SVC (gamma=auto) Model on MNIST Test Set

$$\begin{bmatrix}
594 & 3 & 358 & 18 & 287 & 218 & 69 & 33 & 6 & 414 \\
59 & 601 & 83 & 131 & 311 & 213 & 55 & 510 & 20 & 17 \\
138 & 31 & 1344 & 61 & 55 & 194 & 67 & 73 & 22 & 14 \\
81 & 4 & 151 & 1131 & 16 & 494 & 5 & 73 & 27 & 18 \\
15 & 76 & 72 & 13 & 1212 & 233 & 17 & 188 & 68 & 106 \\
84 & 21 & 147 & 109 & 25 & 1471 & 61 & 50 & 22 & 10 \\
188 & 9 & 431 & 22 & 121 & 411 & 795 & 5 & 8 & 10 \\
49 & 239 & 447 & 248 & 59 & 420 & 15 & 457 & 43 & 23 \\
75 & 28 & 190 & 184 & 94 & 1012 & 97 & 41 & 249 & 30 \\
28 & 191 & 206 & 233 & 236 & 163 & 12 & 502 & 226 & 203
\end{bmatrix}$$

Figure 10: Confusion Matrix by Gaussian SVC (gamma=auto) Model on USPS Test Set

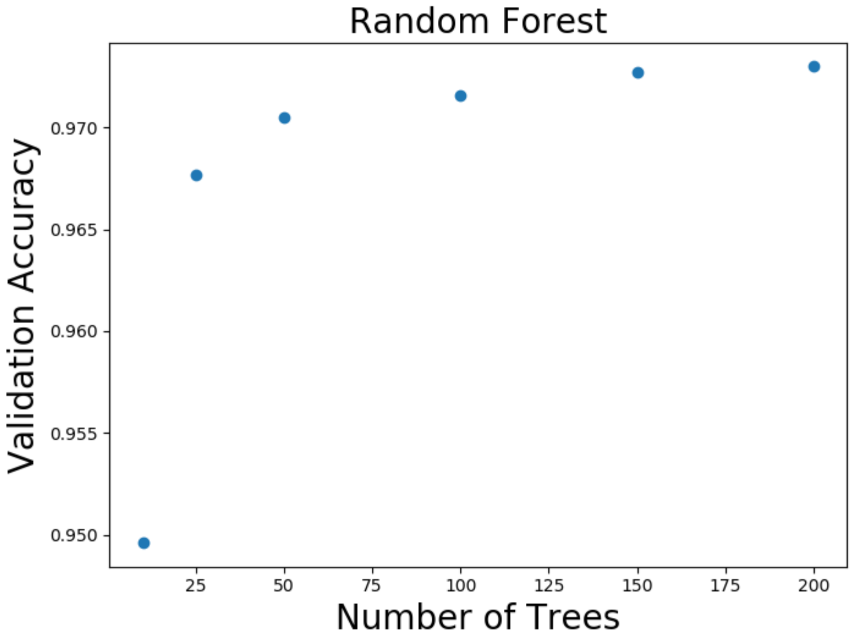


Figure 11: Random Forest Model Performances with Different Number of Trees

seen as the accuracy increases to more than 0.4%. However, that is still a very poor performance, which is as expected.

3.4 Deep Neural Networks

In this section, the results form a two layer deep neural networks model are presented. The model was created via the Keras library. Two hyperparameters were tuned and optimized. The dropout rate was chosen among $\{0.1, 0.2, 0.3, 0.4\}$ and the number of nodes in hidden layers has possible

Table 5: Performance of Random Forest Model

DataSets	Accuracy
MNIST_Training	1.0
MNIST_Validation	0.9733
MNIST_Test	0.9701
USPS_Test	0.41482074103705185

$$\begin{bmatrix} 970 & 0 & 1 & 0 & 0 & 2 & 3 & 1 & 3 & 0 \\ 0 & 1121 & 2 & 4 & 0 & 2 & 3 & 0 & 2 & 1 \\ 6 & 0 & 1000 & 5 & 3 & 0 & 4 & 8 & 6 & 0 \\ 1 & 0 & 10 & 971 & 0 & 8 & 0 & 9 & 8 & 3 \\ 1 & 0 & 2 & 0 & 955 & 0 & 4 & 0 & 2 & 18 \\ 3 & 0 & 0 & 13 & 4 & 855 & 8 & 1 & 5 & 3 \\ 7 & 3 & 0 & 0 & 1 & 3 & 942 & 0 & 2 & 0 \\ 1 & 3 & 17 & 2 & 1 & 0 & 0 & 993 & 2 & 9 \\ 4 & 0 & 6 & 7 & 2 & 7 & 3 & 3 & 932 & 10 \\ 6 & 5 & 3 & 8 & 8 & 2 & 1 & 5 & 9 & 962 \end{bmatrix}$$

Figure 12: Confusion Matrix by Random Forest Model on MNIST Test Set

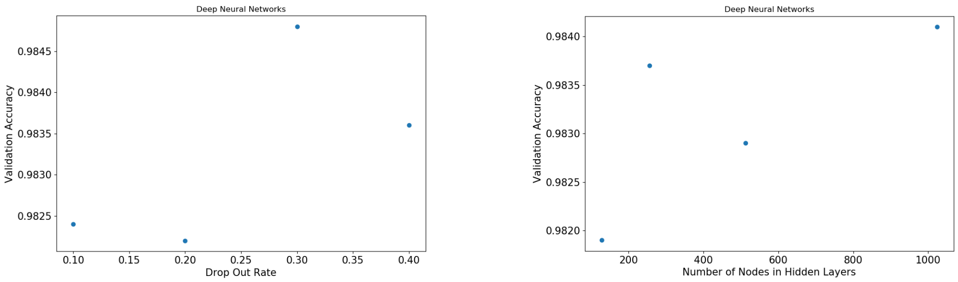
$$\begin{bmatrix} 596 & 22 & 259 & 65 & 467 & 159 & 68 & 108 & 0 & 256 \\ 10 & 713 & 32 & 87 & 15 & 127 & 35 & 979 & 1 & 1 \\ 72 & 43 & 1260 & 75 & 60 & 190 & 12 & 278 & 6 & 3 \\ 40 & 9 & 76 & 1282 & 51 & 330 & 1 & 188 & 3 & 20 \\ 8 & 221 & 36 & 19 & 1092 & 177 & 14 & 384 & 26 & 23 \\ 68 & 39 & 60 & 76 & 16 & 1587 & 13 & 129 & 5 & 7 \\ 270 & 60 & 221 & 23 & 103 & 398 & 803 & 109 & 2 & 11 \\ 27 & 358 & 357 & 257 & 39 & 227 & 29 & 697 & 4 & 5 \\ 42 & 40 & 151 & 216 & 109 & 1096 & 60 & 106 & 165 & 15 \\ 11 & 294 & 210 & 246 & 260 & 129 & 13 & 653 & 83 & 101 \end{bmatrix}$$

Figure 13: Confusion Matrix by Random Forest Model on USPS Test Set

values of $\{128, 256, 512, 1024\}$. The comparison of models with those parameters is shown in fig 14. The optimum parameters are: dropout rate $= 0.3$ nodes $= 1024$. The model with best parameters was evaluated on all three MNIST data sets as well as the USPS test set. Fig 15 and table 6 show the performance of the model. Fig 16 and fig 17 are the confusion matrices on MNIST and USPS test sets, respectively. The deep neural networks model achieved a very high accuracy ($\sim 98\%$) for MNIST datasets. For the USPS test set, the accuracy further improved to more than 50%. However, it is still much lower than the MNIST set accuracies. So the "No Free Lunch" theorem is still unbeaten even for neural networks.

3.5 Convolutional Neural Networks

The covolutional neural networks model is very good in dealing with image recognition. The CNN model was also created via the Keras library. Two hyperparameters tuned were the number of filters and the dropout rates. The dropout rate was chosen among $\{0.1, 0.2, 0.3, 0.4\}$ and the number of



(a) Dropout Rates

(b) Number of Nodes in Hidden Layers

Figure 14: Deep Neural Networks

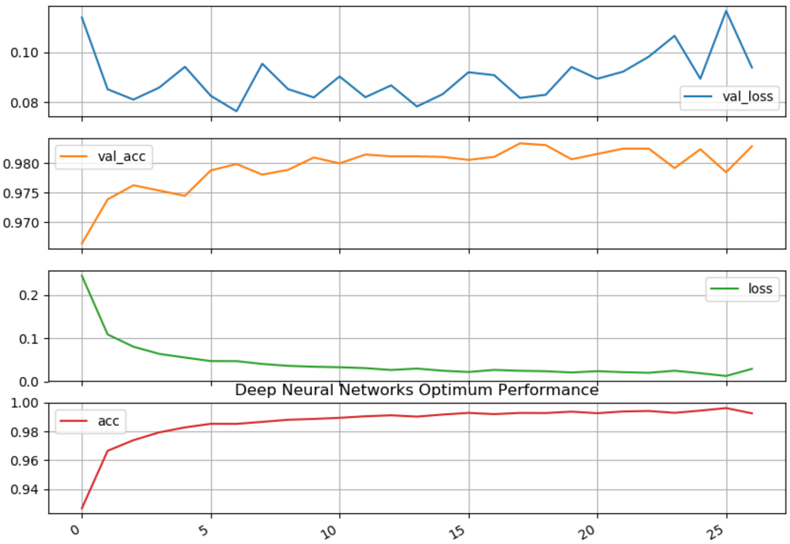


Figure 15: Deep Neural Networks Performance

Table 6: Performance of DNN Model

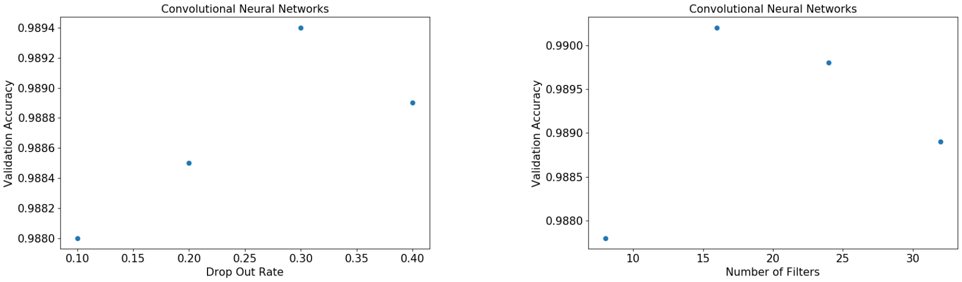
DataSets	Accuracy
MNIST_Training	0.99902
MNIST_Validation	0.9815
MNIST_Test	0.9823
USPS_Test	0.5378768938446923

$$\begin{bmatrix}
969 & 2 & 2 & 0 & 1 & 0 & 3 & 1 & 2 & 0 \\
0 & 1129 & 1 & 3 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1017 & 3 & 1 & 0 & 0 & 7 & 3 & 0 \\
0 & 0 & 4 & 998 & 0 & 2 & 0 & 2 & 2 & 2 \\
1 & 2 & 2 & 0 & 956 & 0 & 6 & 5 & 0 & 10 \\
2 & 0 & 0 & 18 & 1 & 863 & 2 & 0 & 3 & 3 \\
0 & 2 & 0 & 1 & 1 & 4 & 948 & 0 & 2 & 0 \\
0 & 4 & 9 & 4 & 1 & 0 & 0 & 1004 & 1 & 5 \\
0 & 1 & 4 & 7 & 2 & 2 & 0 & 3 & 952 & 3 \\
0 & 3 & 0 & 3 & 4 & 4 & 1 & 3 & 4 & 987
\end{bmatrix}$$

Figure 16: Confusion Matrix by DNN Model on MNIST Test Set

$$\begin{bmatrix}
623 & 11 & 161 & 131 & 234 & 32 & 99 & 33 & 62 & 614 \\
33 & 827 & 70 & 100 & 332 & 15 & 36 & 492 & 30 & 65 \\
47 & 20 & 1671 & 71 & 24 & 34 & 26 & 68 & 29 & 9 \\
14 & 17 & 139 & 1614 & 5 & 142 & 2 & 17 & 39 & 11 \\
3 & 201 & 54 & 27 & 1254 & 28 & 24 & 237 & 130 & 42 \\
42 & 4 & 38 & 132 & 14 & 1545 & 14 & 30 & 162 & 19 \\
69 & 37 & 240 & 20 & 50 & 83 & 1425 & 7 & 57 & 12 \\
23 & 196 & 383 & 455 & 31 & 21 & 24 & 742 & 111 & 14 \\
97 & 45 & 218 & 449 & 85 & 171 & 104 & 77 & 734 & 20 \\
3 & 145 & 135 & 291 & 193 & 19 & 3 & 610 & 279 & 322
\end{bmatrix}$$

Figure 17: Confusion Matrix by DNN Model on USPS Test Set



(a) Dropout Rates

(b) Number of Filters

Figure 18: Convolutional Neural Networks

filters has possible values of $\{8, 16, 24, 32\}$. The comparison of models with those parameters is shown in fig 18. The optimum parameters are: dropout rate $= 0.3$ and number of filters $= 16$. The model with best parameters was evaluated on all three MNIST data sets as well as the USPS test set. Fig 19 and table 7 show the performance of the model. Fig 20 and fig 21 are the confusion matrices on MNIST and USPS test sets, respectively. The convolutional neural networks model got the highest accuracy (more than 99%) for MNIST datasets. The same trend can be observed for the USPS data set as its accuracy reached the highest by the predictions from CNN. However, it is still lower than 60%.

3.6 Combining Models

The models in this project were combined by two different methods: direct hard voting and hard voting after boosting. In a hard voting system, the prediction for every image was made by a simple majority voting across all five classifiers. The prediction accuracy by this method is 0.9782 for the MNIST test set and 0.4973248662433122 for the USPS test set, respectively. The confusion matrices for these sets are shown in Fig 22 and fig 23. It is observed that the accuracy for the combined model is lower than that from the convolutional neural networks model. The possible reason for that is the other models are very inaccurate compared to neural networks models. So a direct hard voting process could not make a better model. With the boosting approach, the five models were trained on 60% of the training set in sequence. The data points that were misclassified by previous models

Table 7: Performance of CNN Model

DataSets	Accuracy
MNIST_Training	0.99872
MNIST_Validation	0.9902
MNIST_Test	0.99
USPS_Test	0.5784289214460723

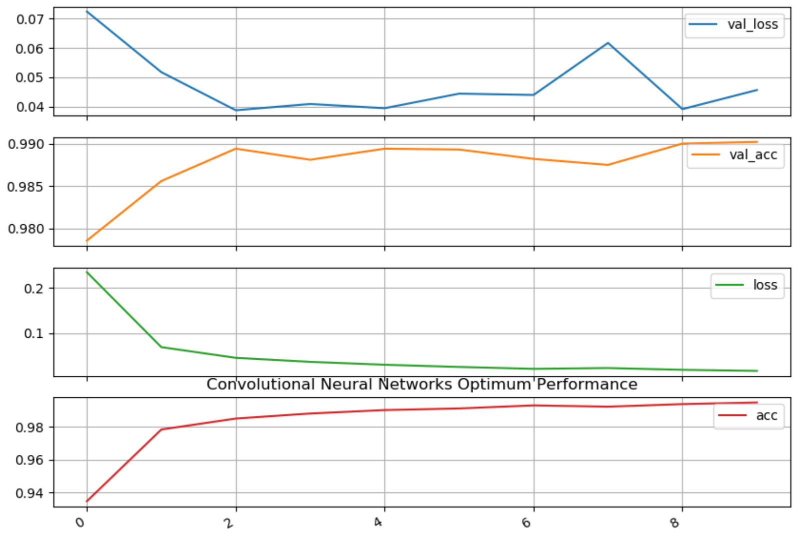


Figure 19: Convolutional Neural Networks Performance

$$\begin{bmatrix}
976 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 1 & 0 \\
0 & 1130 & 2 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\
2 & 0 & 1026 & 0 & 0 & 0 & 0 & 3 & 1 & 0 \\
0 & 0 & 1 & 1004 & 0 & 2 & 0 & 1 & 1 & 1 \\
2 & 0 & 1 & 0 & 966 & 0 & 4 & 0 & 1 & 8 \\
1 & 0 & 0 & 7 & 0 & 881 & 2 & 0 & 0 & 1 \\
6 & 2 & 0 & 1 & 1 & 3 & 945 & 0 & 0 & 0 \\
0 & 3 & 6 & 2 & 0 & 0 & 0 & 1016 & 1 & 0 \\
2 & 0 & 0 & 3 & 0 & 1 & 0 & 1 & 965 & 2 \\
3 & 1 & 0 & 1 & 3 & 2 & 0 & 3 & 4 & 992
\end{bmatrix}$$

Figure 20: Confusion Matrix by CNN Model on MNIST Test Set

$$\begin{bmatrix}
971 & 8 & 108 & 89 & 106 & 22 & 37 & 13 & 123 & 523 \\
153 & 829 & 180 & 36 & 384 & 32 & 75 & 285 & 15 & 11 \\
32 & 9 & 1680 & 109 & 25 & 34 & 6 & 20 & 84 & 0 \\
5 & 2 & 37 & 1724 & 2 & 208 & 2 & 4 & 15 & 1 \\
3 & 33 & 43 & 36 & 1437 & 26 & 12 & 175 & 220 & 15 \\
5 & 1 & 32 & 267 & 4 & 1610 & 0 & 30 & 22 & 29 \\
103 & 9 & 311 & 40 & 25 & 86 & 1390 & 2 & 30 & 4 \\
15 & 57 & 383 & 575 & 17 & 37 & 12 & 861 & 43 & 0 \\
20 & 8 & 93 & 463 & 30 & 338 & 8 & 56 & 921 & 63 \\
1 & 17 & 206 & 515 & 93 & 15 & 1 & 443 & 316 & 393
\end{bmatrix}$$

Figure 21: Confusion Matrix by CNN Model on USPS Test Set

$$\begin{bmatrix}
972 & 0 & 1 & 1 & 0 & 1 & 2 & 1 & 2 & 0 \\
0 & 1127 & 2 & 2 & 0 & 0 & 3 & 0 & 1 & 0 \\
3 & 0 & 1012 & 3 & 1 & 0 & 1 & 7 & 5 & 0 \\
0 & 0 & 7 & 988 & 0 & 4 & 0 & 3 & 6 & 2 \\
2 & 0 & 1 & 0 & 963 & 0 & 4 & 0 & 2 & 10 \\
4 & 0 & 0 & 13 & 0 & 863 & 3 & 1 & 5 & 3 \\
7 & 2 & 1 & 2 & 1 & 6 & 939 & 0 & 0 & 0 \\
0 & 4 & 14 & 2 & 1 & 0 & 0 & 997 & 1 & 9 \\
4 & 0 & 4 & 7 & 2 & 4 & 1 & 2 & 946 & 4 \\
6 & 5 & 1 & 8 & 6 & 1 & 1 & 4 & 2 & 975
\end{bmatrix}$$

Figure 22: Confusion Matrix by the Combined Model (Hard Voting) on MNIST Test Set

$$\begin{bmatrix}
766 & 10 & 241 & 67 & 234 & 74 & 38 & 26 & 48 & 496 \\
89 & 784 & 64 & 202 & 245 & 63 & 22 & 492 & 28 & 11 \\
105 & 19 & 1623 & 67 & 34 & 68 & 11 & 42 & 22 & 8 \\
43 & 11 & 122 & 1555 & 5 & 203 & 1 & 26 & 25 & 9 \\
13 & 124 & 36 & 34 & 1274 & 73 & 6 & 227 & 166 & 47 \\
68 & 12 & 71 & 159 & 16 & 1563 & 9 & 31 & 60 & 11 \\
213 & 30 & 398 & 38 & 76 & 216 & 1005 & 1 & 11 & 12 \\
53 & 224 & 350 & 486 & 29 & 122 & 12 & 604 & 108 & 12 \\
126 & 20 & 159 & 371 & 80 & 601 & 41 & 53 & 527 & 22 \\
15 & 162 & 147 & 419 & 127 & 56 & 3 & 534 & 292 & 245
\end{bmatrix}$$

Figure 23: Confusion Matrix by the Combined Model (Hard Voting) on USPS Test Set

have a five times higher percentage to be selected for the next training. A hard voting progress was then applied to the trained models to get the combined predictions. The prediction accuracy by the boosting method is 0.9807 for the MNIST test set and 0.4986249312465623 for the USPS test set, respectively. Fig 24 and fig 25 are the confusion matrices by this combined model on the MNIST and USPS test sets, respectively. This is a slight improvement than the direct hard voting method, but it is still less accurate than the CNN model. That is possibly because that the CNN model is already extremely accurate and it is very difficult to improve it by combing it with other less accurate models.

4 Output

There are two output files for this project: output_MNIST.csv, output_USPS.csv and confusion_matrix.dat. In output_MNIST.csv file, there are nine columns. From left to right, they are: Row ID, the true labels for the MNIST test set, the labels predicted by logistic regression, the labels predicted by linear SVC, the labels predicted by random forest, the labels predicted by deep

$$\begin{bmatrix}
971 & 0 & 0 & 1 & 0 & 0 & 4 & 1 & 3 & 0 \\
0 & 1126 & 3 & 0 & 0 & 1 & 1 & 1 & 3 & 0 \\
4 & 0 & 1013 & 2 & 1 & 0 & 1 & 6 & 5 & 0 \\
0 & 0 & 6 & 987 & 0 & 6 & 0 & 3 & 6 & 2 \\
1 & 1 & 2 & 0 & 957 & 0 & 5 & 0 & 2 & 14 \\
2 & 0 & 0 & 9 & 0 & 872 & 4 & 1 & 2 & 2 \\
5 & 3 & 1 & 1 & 3 & 8 & 936 & 0 & 1 & 0 \\
2 & 4 & 13 & 2 & 0 & 0 & 0 & 998 & 2 & 7 \\
3 & 0 & 3 & 4 & 4 & 3 & 0 & 2 & 953 & 2 \\
6 & 6 & 0 & 5 & 4 & 2 & 0 & 2 & 1 & 983
\end{bmatrix}$$

Figure 24: Confusion Matrix by the Combined Model (Boosting) on MNIST Test Set

$$\begin{bmatrix}
709 & 5 & 242 & 34 & 170 & 79 & 47 & 20 & 46 & 648 \\
36 & 672 & 97 & 210 & 234 & 127 & 20 & 513 & 74 & 17 \\
80 & 22 & 1638 & 58 & 21 & 110 & 10 & 33 & 18 & 9 \\
29 & 4 & 140 & 1448 & 5 & 310 & 1 & 26 & 26 & 11 \\
19 & 84 & 43 & 42 & 1186 & 107 & 8 & 220 & 228 & 63 \\
41 & 15 & 71 & 93 & 15 & 1669 & 6 & 26 & 50 & 14 \\
224 & 23 & 389 & 24 & 67 & 250 & 970 & 5 & 26 & 22 \\
44 & 209 & 403 & 480 & 22 & 125 & 13 & 542 & 155 & 7 \\
137 & 22 & 129 & 290 & 48 & 661 & 46 & 46 & 585 & 36 \\
12 & 144 & 155 & 396 & 126 & 61 & 4 & 435 & 372 & 295
\end{bmatrix}$$

Figure 25: Confusion Matrix by the Combined Model (Boosting) on USPS Test Set

neural networks the labels predicted by convolutional neural networks, the labels predicted by the combined model (hard voting) and the labels predicted by the combined model (boosting). The output_USPS.csv file has the same structure except that now the predictions are for the USPS test set. The confusion_matrix.dat contains the confusion matrices for both MNIST and USPS test sets from all the models.