# Digit Recognizer

Zhifei Zhang
Department of Electrical Engineering and Computer Science, UTK
zzhang61@vols.utk.edu

Shiqi Zhong
Department of Electrical Engineering and Computer Science, UTK
szhong4@vols.utk.edu

Liang Tong
Department of Electrical Engineering and Computer Science, UTK
ltong@vols.utk.edu

***Abstract*—This project implements multiple classifiers on the well-know handwritten digit database MNIST. Exactly, MPP, kNN, BPNN, decision tree, kmeans, WTA, SOM, multi-SVM, random forest and CNN are applied. In addition, three classifier fusion methods—majority voting, confusion matrix and BKS—are used to improve classification performance. In experiment results, accuracy and computation time are shown to illustrate the performance of each classifier. Currently, the error rate of digit recognition on MNIST database is 0.23%, which is not easy to be surpassed. Thus, the purpose of this project is to make a deeper understanding on different classifiers.**

## I. Introduction

Handwritten digit recognition is a sort of well studied topic in pattern recognition and machine learning research. Its history can be tracked back to around three decades ago. In 1984, [1] presented a hierarchical structural approach to recognize certain formula, and experiment was performed on a small-size data set (40 samples). Finally, it got an accuracy of 98.1%. Until now, a lot of researchers have competed to increase the accuracy rate on those well-know handwritten digit database MNIST. Most recently, the error rate can be reduced to 0.23% using convolutional neural network (CNN) [2]. It is an exciting achievement compared to the first shot [3]–[9] in 1998, when the error rate was 0.7% using CNN as well. Although handwritten digit recolonization is an "old" problem, it is becoming an increasingly challenge topic as the error rate approaches zero.

The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems [10]. This database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image [11]. Fig. 1 shows some samples from MNIST training data set. Since the first shot on the MNIST database, multiple algorithms are applied on it, such as linear classifier, kNN, decision tree, SVM, neural network, CNN, etc.

The best result using linear classifier is 7.6% [12], in which the data is preprocessed by deskewing, and pairwise linear classifier is applied. [13] reduces the error rate to 0.52% using kNN with non-linear deformation. Boosted stumps cooperated with Haar features is employed in [14], and the error rate is 0.87%. For SVM, [15] achieves the lowest error rate of 0.56%. In this work, virtual SVM is presented to get better support vector. In recent years, the lowest error rate record is refreshed again and again as the development of neural network and deep learning (e.g. CNN). [16] reduces the record to 0.35% using neural network. Then, [2] refreshes the record to 0.23% using CNN and still keep the best record since 2012. Table I shows the previous results on MNIST database.

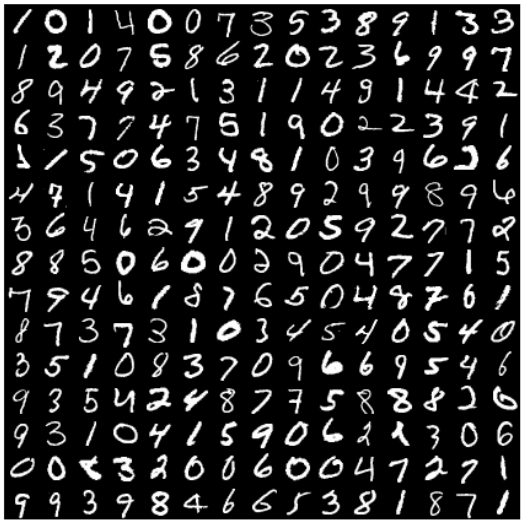


Fig. 1. Samples of handwritten digit

TABLE I
Previous results on MNIST database

| Classifier | Preprocessing | Error rate (%) |
|---|---|---|
| Pairwise linear classifier | Deskewing | 7.6 [12] |
| Boosted Stumps | Haar features | 0.87 [14] |
| SVM | Deskewing | 0.56 [15] |
| kNN | Shiftable edges | 0.52 [13] |
| Neural Network | None | 0.35 [16] |
| CNN | Width normalizations | 0.23 [2] |

## II. Subset of MNIST

The original MNIST database has 60,000 training samples and 10,000 testing samples. It will be really time-consuming if a relatively complex algorithm is applied. In [2], for instance, CNN is implemented with 800 iterations, which may cost tens of hours on a common computer. For the purpose of implementing different classification methods and comparing their performance, a subset of MNIST is used. First, we get a subset (48,000 samples) from Kaggle—Digit Recognizer.

Then, one out of ten of the Kaggle data is kept in our final subset. Thereby, there are totally 4800 samples in our processing.

## III. Classifiers

In this project, several classifiers are implemented, which are listed in lecture order as following:

- Maximum Posterior Probability (MPP)
- K Nearest Neighbors (kNN)
- Back-Propagation Neural Network (BPNN)
- k-means and winner-take-all
- Self-Organizing Map (SOM)
- Decision Tree (DT)
- Support Vector Machine (SVM)
- Random Forest (RF)
- Convolutional Neural Network (CNN)

Each classifier is applied on the same subset from MNIST, and cross validation is employed to derive. In detail, 20% of the subset is randomly selected as testing data, and the remain is treated as training data. Then, a classifier is applied on the training and testing data. This procedure can be iterated several times to ensure a stable accuracy.

### A. Maximum Posterior Probability (MPP)

Maximum Posterior Probability(MPP) is based on Bayesian rule. The classifier for MPP is the discriminant function $g_i(x)$. Consider a two-class case (equal prior), $g_1(x) = P(w_1|x)$ and $g_2(x) = P(w_2|x)$. For a given $x$, if $P(w_1|x) > P(w_2|x)$, $x$ belongs to class 1, otherwise $x$ belongs to class 2. There are three different discriminant functions based on different assumptions. The pre-assumption for 3 cases is that each class obeys Gaussian distribution. Case 1 is a linear discriminant function which assumes that all features are statistically independent and have the same variance. Case 2 is also a linear discriminant function, which assumes that covariance matrices for all the classes are identical but not a scalar of identity matrix. Case 3 is a hyperquadratic discriminant function, and it has no assumption on the covariance matrices for each category.

### B. K Nearest Neighbors (kNN)

The basic concept of kNN is quite simple. Given a test sample $x$, which needs to be classified, a hypersphere with volume $V$ can be found, in which $k$ training samples are enclosed. If the number of training samples of class $m$ is larger than any other enclosed sample classes, then the test sample will be categorized as class $m$. If there are equal number of samples with two classes, then choose the class which has the minimum distance to the test sample.

Although this way seems very intuitive, it is an equivalent way of MAP. Suppose there are $n$ samples in total, while $n_m$ samples for each class $m$. For the enclosed $k$ samples, there are $k_m$ samples in class $m$. The component of calculating MAP can be represented as below:

- prior probability: $P(\omega_m) = \dfrac{n_m}{n}$
- probability density function: $p(x|\omega_m) = \dfrac{k_m}{n_m V}$
- normalization constant: $p(x) = \dfrac{k}{nV}$

Thus MAP can be finally calculated as:

$$p(\omega_m|x) = \frac{p(x|\omega_m)P(\omega_m)}{p(x)} = \frac{\frac{k_m}{n_m V}\frac{n_m}{n}}{\frac{k}{nV}} = \frac{k_m}{k} \tag{1}$$

### C. Back-Propagation Neural Network (BPNN)

Back-Propagation Neural Network (BPNN) is a feed forward neural network which means there is no feedback during operation, and the back-propagation only processes during determination of weights. The general algorithm flow is shown as follows:

```
Initialize network weights (often small random values)
repeat
    for all training samples do
        feed forward
        compute error
        update weights
    end for
until certain condition is met
```

After initializing the parameters for the neural network, we can use the BPNN classifier.

### D. K-means and Winner-Take-All (WTA)

Clustering is an unsupervised method to classify a dataset. In this project, k-meas and winner-take-all (WTA) are applied. The k-means algorithm is shown as follows:

```
Initialize clusters
repeat
    assign all samples to their nearest clusters
    compute the sample mean of each cluster
until classification of all samples does not change
```

The WTA algorithm:

```
Initialize clusters
repeat
    for each sample do
        assign the sample to the nearest cluster
        update the center of the corresponding cluster
    end for
until classification of all samples converges
```

### E. Self-Organizing Map (SOM)

This approach is also known as Kohonen Feature Maps, which is a kind of neural network approach. It is like winner-take-all approach except that after each sample is classified, not only the winner will be pulled toward sample, but other samples are also pulled toward the sample, with different weights. Its procedure is like:

```
assign each cluster with a random value of cluster center
repeat
    for each sample x do
        find the nearest cluster center ω_α
```

5: modify $\omega_\alpha$ with the following formula:

$$\boldsymbol{\omega}_r^{k+1} = \boldsymbol{\omega}_r^k + \epsilon(k)\Phi(k)(\boldsymbol{x} - \boldsymbol{\omega}_r^k)$$

where

$$\epsilon(k) = \epsilon_{max}(\frac{\epsilon_{min}}{\epsilon_{max}})^{\frac{k}{k_{max}}}$$

$$\Phi(k) = exp(-\frac{\|g_{\boldsymbol{\omega}_r} - g_{\boldsymbol{\omega}_{winner}}\|^2}{2\sigma^2})$$

6: **end for**
7: **until** classification of all samples converges
8: use $\boldsymbol{\omega_\alpha}$ as the cluster center

### F. Decision Tree (DT)

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represents classification rules.

In decision analysis a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of 3 types of nodes:

1) Decision nodes - commonly represented by squares.
2) Chance nodes - represented by circles.
3) End nodes - represented by triangles.

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal. If in practice decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

### G. Support Vector Machine (SVM)

Digit recognition is a multi-class problem, but normal SVM can only separate two classes. So, one v.s. all method is applied here to realize a multi-SVM classifier. The object function of SVM with soft margin can be written in Eq. 2.

$$\begin{aligned} \arg\min_{\mathbf{w},\xi,b} &= \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \right\} \\ s.t.\ \ & y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \le 1 - \xi_i, \\ & \xi_i \le 0,\ i = 1, \cdots, n \end{aligned} \tag{2}$$

where the $\xi_i$ is non-negative slack variables that allow points to be on the wrong side of their soft margin, as well as the decision boundary, and C is a cost parameter that controls the amount of overlap [17].

One v.s. all is to set one class as class 1 and the others as class 2, then a SVM classifier is trained using the redefined dataset. So, there will be 10 different SVM classifiers since there are 10 digits (0∼9). In prediction process, a new sample will be tested on each classifier, and each classifier will yield a estimated label and corresponding probability (confidence of estimation). Finally, the estimation with the highest confidence will be accepted as the finally output of the multi-SVM classifier.

### H. Random Forest (RF)

In decision tree, a tree growing very deep tends to learn highly irregular patterns, which makes it over fit the training set because it has low bias, but very high variance. Random forest is a way of averaging multiple deep decision trees, trained on different subsets of the same training set, with the goal of reducing the variance [18]. But the tree learning process is a little bit different in random forest: if one or a few features are very strong predictors for the label, these features will be selected in many of the trees, causing them to become correlated [19].

Basically, each tree in the random forest is set up from a randomly selected subset of the raw training set. For example, there are $n$ trees, each of which is trained on a random subset and becomes a decision tree $f_i$ $(i = 1, \cdots, n)$. In prediction process, the label of a new sample is given by averaging the predictions from all the individual decision trees. Eq. 3 shows the decision scheme.

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{3}$$

where $x$ denotes a new testing sample.

### I. Convolutional Neural Network (CNN)

A convolutional neural network is a type of feed-forward and back-propagation neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field [20]. Convolutional networks were inspired by biological processes and are variations of multilayer perceptron which is designed to use minimal amounts of preprocessing [21]. They are widely used models for image recognition. A common CNN model (LeNet model) is shown in Fig. 2.

## IV. Experiment result

Each item of the raw data is a vectorized $28 \times 28$ gray image. So, each sample is a $28 \times 28 = 784$ dimensional vector. For some classifiers, such as SVM and random forest, the sample vector can be used directly as features. But in CNN, for example,the sample vectors have to be reconstructed into images. If we use the sample vectors as feature space directly, dimension reduction is necessary. In this project, correlate coefficients are employed to relatively less correlated features. Practically, a feature with P value larger than 0.05 is considered as uncorrelated. Finally, the number of features can be reduced from 784 to around 440. In addition, if PCA is applied with 95% information kept, there will be around 180 features left. However, PCA will decrease accuracy in experiment. So, we only used around 440 features finally in order to balance accuracy and computation time. In addition,

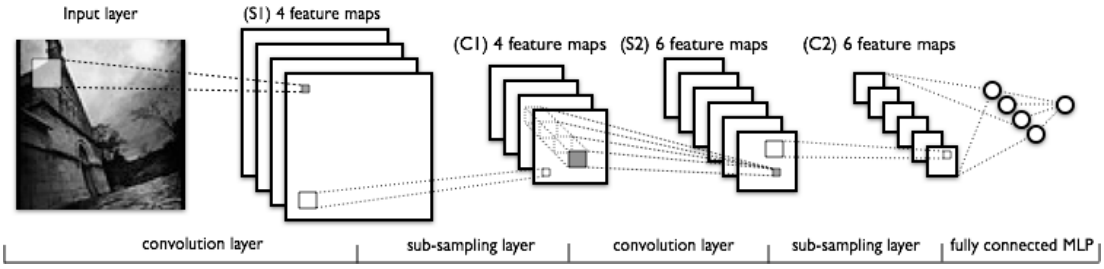



Fig. 2. CNN model [20]

leave-20%-out cross validation is iterated 5 times for each classifier.

### A. MPP

To perform the 3 cases of discriminant function, the build-in function in the Matlab was used. As for the experiment, all of the 3 cases were used to evaluate the accuracy and computation time. The result is shown in Fig. 3 and Table II.

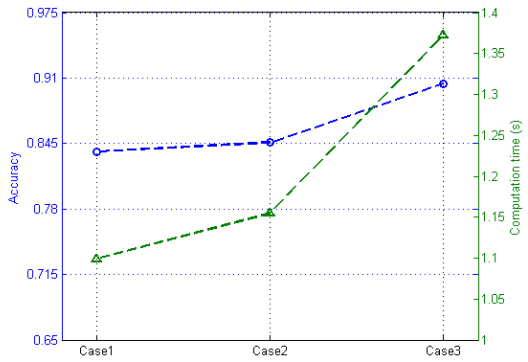



Fig. 3. Experiment result of MPP (○: accuracy; △: computation time)

TABLE II
EXPERIMENT RESULT OF MPP

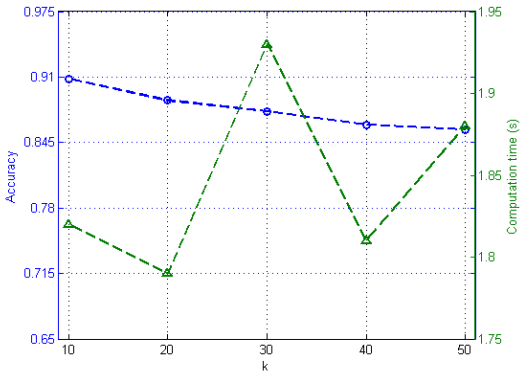
| Case | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 1 | 0.8369 | 0.0055 | 1.0992 |
| 2 | 0.8460 | 0.0018 | 1.1547 |
| 3 | 0.9045 | 0.0052 | 1.3727 |

### B. kNN

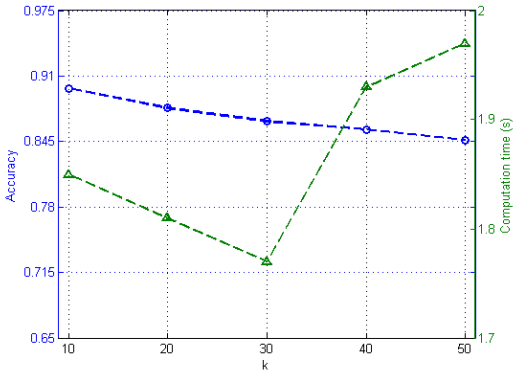
Matlab build-in function is called to implement kNN. we use both *Euclidean Distance* and *City-block Distance* as the distance between two clusters. Fig. 4 and Table III indicate that when $k$ increases, the accuracy will decrease, also those two distances have very closed performance when $k$ is the same.

### C. BPNN

The Matlab package named DeepLearnToolbox [22] is used to implement BPNN. One hidden layer is used in BPNN. After dimension reduction, the dimension of each sample is 441, so the input layer of BPNN has 441 nodes. There are 10 different digits, so the output layer has 10 nodes. For the hidden layer, more nodes means more time-consuming. Here, we set around 60 nodes for the hidden layer. Fig. 5 and Table IV show the experiment result.




(a) Euclidean distance




(b) City-block distance

Fig. 4. Experiment result of kNN (○: accuracy; △: computation time)

### D. K-means and WTA

To perform the experiment, the build-in k-means function in Matlab is used, and different kinds of distances are utilized. In k-means, the total cluster number is set as the number of classes (10). After clustering, the samples were divided into 10 clusters but the label for each cluster does not match the ground truth. Therefore, we check the ground truth of all

TABLE III
EXPERIMENT RESULT OF KNN

(a) *Euclidean Distance*

| k | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 10 | 0.9083 | 0.0048 | 1.82 |
| 20 | 0.8867 | 0.0041 | 1.79 |
| 30 | 0.8760 | 0.0043 | 1.93 |
| 40 | 0.8624 | 0.0037 | 1.81 |
| 50 | 0.8581 | 0.0032 | 1.88 |

(b) *City-block Distance*

| k | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 10 | 0.8979 | 0.0038 | 1.85 |
| 20 | 0.8781 | 0.0050 | 1.81 |
| 30 | 0.8648 | 0.0054 | 1.77 |
| 40 | 0.8571 | 0.0036 | 1.93 |
| 50 | 0.8464 | 0.0052 | 1.97 |

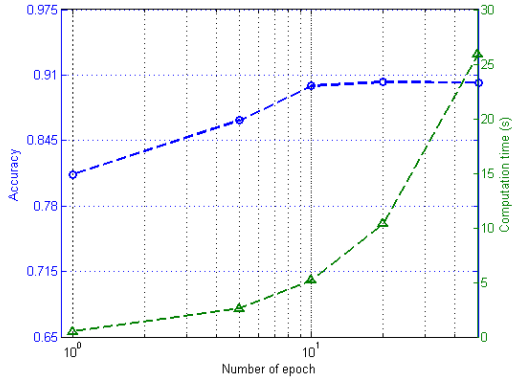



Fig. 5. Experiment result of BPNN (○: accuracy; △: computation time)

TABLE IV
EXPERIMENT RESULT OF BPNN

| Number of epoch | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 1 | 0.8114 | 0.0311 | 0.52 |
| 5 | 0.8650 | 0.0130 | 2.59 |
| 10 | 0.8993 | 0.0055 | 5.18 |
| 20 | 0.9029 | 0.0053 | 10.36 |
| 50 | 0.9026 | 0.0033 | 25.90 |

samples within a cluster and use the major label as ground truth of this cluster. Table V shows the experiment result.

TABLE V
EXPERIMENT RESULT OF K-MEANS

| Distance | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| City-block | 0.4036 | 0.0107 | 0.9349 |
| Euclidean | 0.4088 | 0.0259 | 0.8624 |
| Cosine | 0.4350 | 0.0185 | 0.7743 |
| Correlation | 0.4548 | 0.0288 | 0.7731 |

By the same token, WTA is implemented using the wta.cpp provides in class. Not surprisingly, the accuracy is 0.4352, which is similar with k-means.

## E. SOM

We use Matlab build-in function to implement SOM by using different epoch iterations. The results are shown in Fig. 6 and Table VI. When the epoch iterations increase, it needs

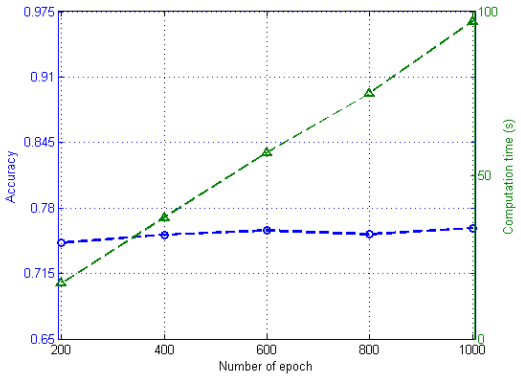



Fig. 6. Experiment result of SOM (○: accuracy; △: computation time)

TABLE VI
EXPERIMENT RESULT OF SOM

| Epoch | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 200 | 0.7455 | 0.0100 | 17 |
| 400 | 0.7533 | 0.0110 | 37 |
| 600 | 0.7576 | 0.0065 | 57 |
| 800 | 0.7538 | 0.0068 | 75 |
| 1000 | 0.7600 | 0.0098 | 97 |

more time to converge. However, the accuracy and standard error are almost the same. When $epoch = 600$, the SOM weight positions are shown in Fig. 7.

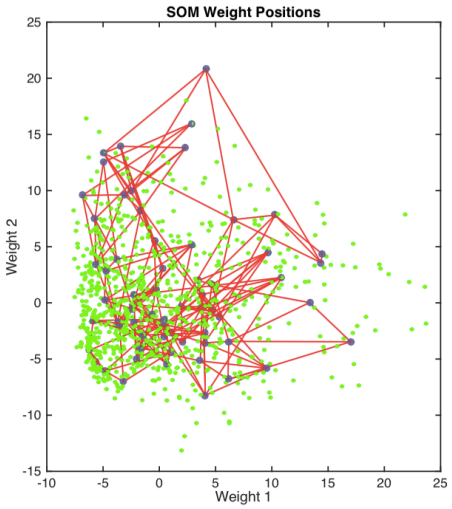
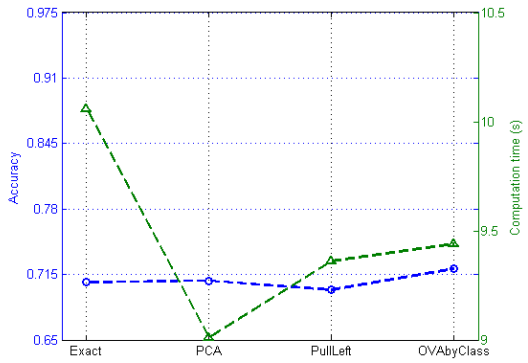


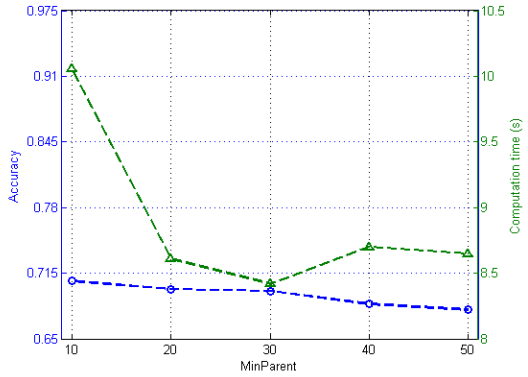

Fig. 7. SOM weight positions

## F. DT

Matlab build-in function is called to implement Decision Tree. Firstly we use 4 different algorithms in MATLAB:*Exact, PCA, PullLeft* and *OVAbyClass*. We then use different *MinParent* (which means minimum number of branch node observations) under the *Exact* algorithm. The results are in Fig.




(a) Different algorithms




(b) Different MinParent

Fig. 8. Experiment result of DT (○: accuracy; △: computation time)

8 and Table VII. For the four algorithms, they can get nearly the same accuracy. However, when we use larger *MinParent*, the accuracy will decrease.

TABLE VII
EXPERIMENT RESULT OF DT

(a) Different Algorithms

| Alglrithm | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| Exat | 0.7074 | 0.0104 | 10.06 |
| PCA | 0.7088 | 0.0046 | 9.01 |
| PullLeft | 0.6998 | 0.0060 | 9.36 |
| OVAbyClass | 0.7205 | 0.0071 | 9.44 |

(a) Different MinParent

| MinParent | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 10 | 0.7074 | 0.0104 | 10.06 |
| 20 | 0.6993 | 0.0077 | 8.61 |
| 30 | 0.6974 | 0.0088 | 8.42 |
| 40 | 0.6845 | 0.0106 | 8.7 |
| 50 | 0.6788 | 0.0108 | 8.65 |

### G. SVM

LibSVM-3.17 [23] is employed to implement basic SVM algorithm, where radial basis function (RBF) is chosen as the kernel. In LibSVM-3.17, the RBF kernel function is expressed as Eq. 4.

$$RBF = \exp\left(-\gamma|u-v|^2\right) \tag{4}$$

where $\gamma$ is equivalent to the inverse of variance. Larger $\gamma$ will cause sharper decision boundary, which may result in over fitting. Experimentally , $\gamma = 1/n$, where $n$ is the number of features. Through experiment, we set $\gamma = 0.001$, under which the highest accuracy can be achieved and the computation time is around 68 seconds. Expatriation result is shown in Fig. 9 and Table VIII. Note that the computation time corresponds to the running time of one out of five iterations in cross validation.

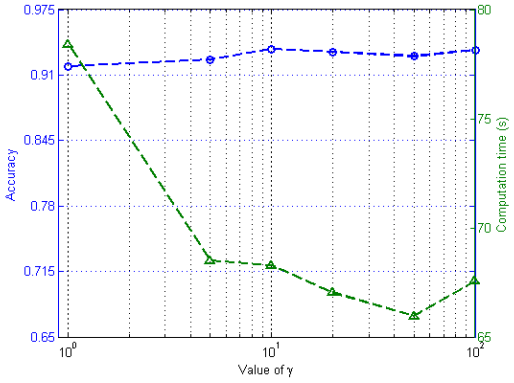



Fig. 9. Experiment result of SVM (○: accuracy; △: computation time)

TABLE VIII
EXPERIMENT RESULT OF SVM

$\gamma = 0.001$

| Cost | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 1 | 0.9186 | 0.0050 | 78.40 |
| 5 | 0.9252 | 0.0034 | 68.50 |
| 10 | 0.9357 | 0.0028 | 68.27 |
| 20 | 0.9331 | 0.0024 | 67.03 |
| 50 | 0.9286 | 0.0027 | 65.96 |
| 100 | 0.9350 | 0.0017 | 67.55 |
| 1000 | 0.9364 | 0.0042 | 64.89 |

cost $= 10$

| $\gamma$ | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 0.01 | 0.8602 | 0.0044 | 395.19 |
| 0.001 | 0.9357 | 0.0028 | 68.27 |
| 0.0001 | 0.8943 | 0.0028 | 68.89 |

### H. RF

Matlab build-in function is called to implement random forest. Fig. 10 shows the result. And Table IX shows the detailed result.

TABLE IX
EXPERIMENT RESULT OF RANDOM FOREST

| Number of trees | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 1 | 0.6624 | 0.0047 | 0.14 |
| 10 | 0.8660 | 0.0027 | 0.81 |
| 100 | 0.9245 | 0.0018 | 7.45 |
| 1000 | 0.9298 | 0.0038 | 73.38 |
| 10000 | 0.9288 | 0.0048 | 884.96 |

Samples of individual trees is shown in Fig. 11.

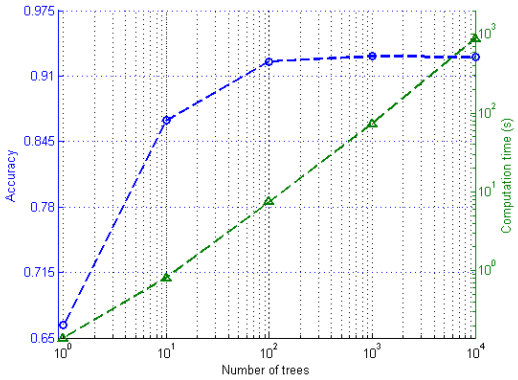



Fig. 10. Experiment result of RF (○: accuracy; △: computation time)

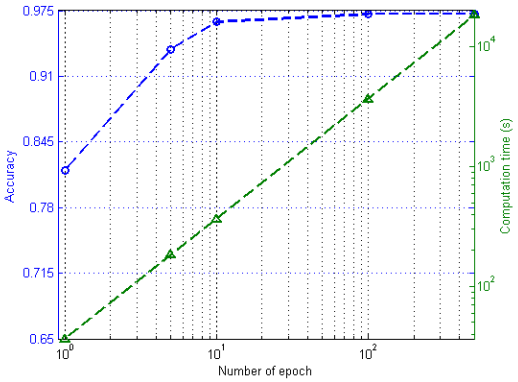
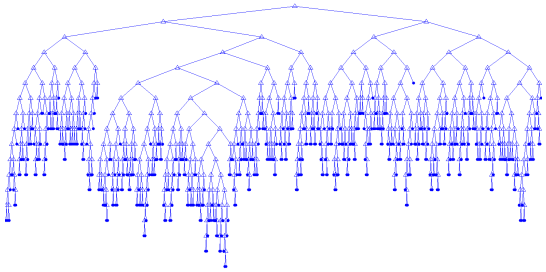
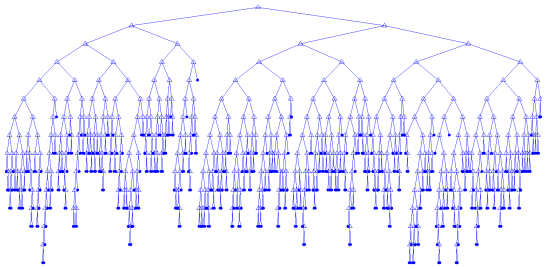


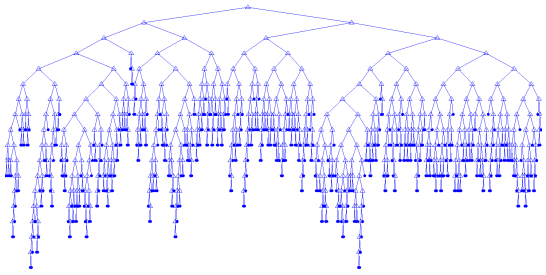

Fig. 12. Experiment result of CNN (○: accuracy; △: computation time)



(a) tree No.1



(b) tree No.2



(c) tree No.10

Fig. 11. Samples of individual trees in RF experiment (totally 10 trees)

### I. CNN

The Matlab package named DeepLearnToolbox [22] is used to implement CNN. Totally, two layers are used in CNN, and the two layer use 6 and 12 filters respectively. In maximum pooling process, $2 \times 2$ widow is used. Fig. 12 and Table X show the result. It can achieve an accuracy of 97.14% at most.

TABLE X
EXPERIMENT RESULT OF CNN

| Number of epoch | Accuracy | Standard error | Computation time (s) |
|---|---|---|---|
| 1 | 0.8167 | 0.0038 | 35 |
| 5 | 0.9369 | 0.0021 | 175 |
| 10 | 0.9637 | 0.0009 | 350 |
| 100 | 0.9714 | 0.0001 | 3500 |
| 500 | 0.9714 | 0.0001 | 17500 |

Compared with the highest accuracy (99.77%) in [2], there are two reasons why we get a lower accuracy:

(1) We use a subset of original MNIST database.
(2) We use less filters and iterations in CNN.

For the first reason, we tried to implement CNN on the original database (60,000 training samples and 10,000 testing samples) with 100 epoch. The accuracy can achieve 98.8%.

For the second reason, [2] runs CNN with 20 and 40 filters respectively for the first and second layers and with 800 epoch. In our project, only 6 and 12 filters are used in corresponding layers, and the number of epoch is 500 at most. In addition, [2] created additional dataset by normalizing digit width to different values and assembles multiple basic CNN to build a larger network.

Above all, smaller data size, simpler structure and less iteration cause the lower accuracy.

## V. CLASSIFIER FUSION

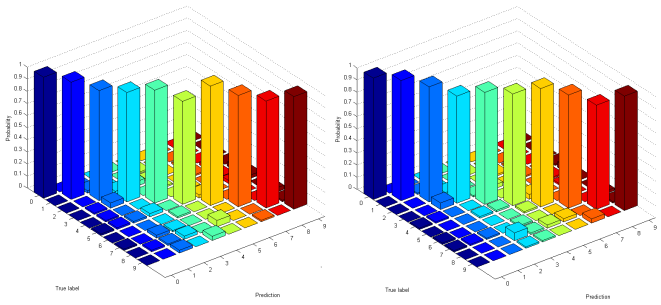
In order to increase accuracy, outputs of multiple classifiers are fused to improve the performance. In this project, three classifier fusion methods are implemented, and three classifiers—RF, SVM and CNN—are fused.

### A. Majority Voting (MV)

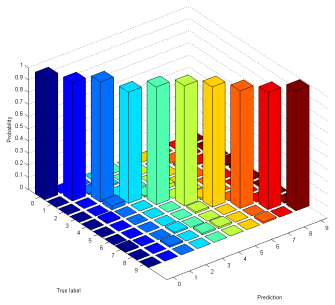
Majority voting is the most direct way to combine different classifiers. A prediction given by most classifier will be considered as the final fused estimation.

### B. Naive Bayes Combination

Naive Bayes combination assumes all classifiers are mutually independent. Confusion matrix (CM) is used here to derive a fused estimation, under which Bayes rule is latent. Fig. 13 shows the confusion matrices of RF, SVM and CNN.



(a) Confusion matrix of RF (b) Confusion matrix of SVM



(c) Confusion matrix of CNN

Fig. 13. Confusion matrices

### C. Behavior-Knowledge Space (BKS)

Behavior-knowledge space method will build a k-dimensional behavior-knowledge space (k is the number of classifiers), which is estimated to accumulate statistical information from the individual classifiers' behaviors derived from learned patterns [24]. After forming the behavior-knowledge space, estimations of all classifiers on a new testing sample can be used as index to find the final decision. Fig. 14 shows the accuracy before and after fusion.

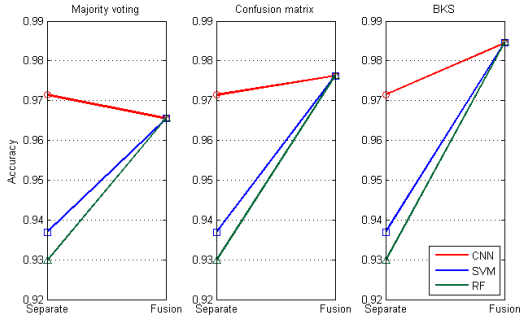



Fig. 14. Accuracy

## VI. Discussion

In this project, different kinds of classifiers are implemented, as well as classifier fusion. Regardless computation time, CNN can achieve the highest accuracy among others, and CNN is more potential to perform better. Implementing classifier fusion with the three classifiers (RF, SVM and CNN), the final accuracy is boosted from 97.1% to 98.5% using BKS method. Overall, the accuracy of different methods is summarized in Table XI and Fig. 15.

TABLE XI
Accuracy of all methods

| Method | Accuracy | Standard error |
|---|---|---|
| WAT | 0.4352 | 0.0292 |
| k-means | 0.4548 | 0.0288 |
| DT | 0.7205 | 0.0071 |
| SOM | 0.7600 | 0.0098 |
| BPNN | 0.9029 | 0.0053 |
| MPP | 0.9045 | 0.0052 |
| kNN | 0.9083 | 0.0048 |
| RF | 0.9298 | 0.0038 |
| SVM | 0.9364 | 0.0042 |
| CNN | 0.9714 | 0.0001 |
| MV | 0.9655 | — |
| CM | 0.9762 | — |
| BKS | 0.9845 | — |

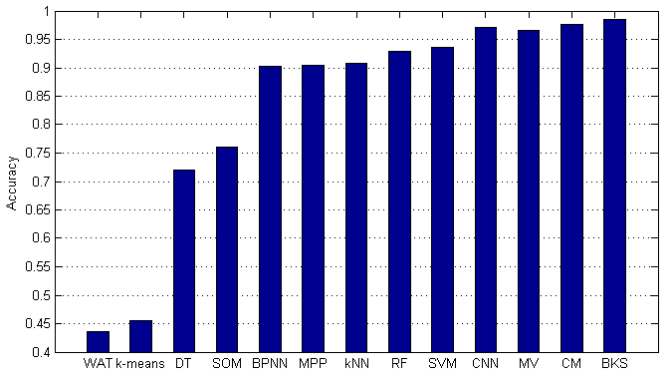



Fig. 15. Accuracy of all methods

From the experiment results, unsupervised methods (k-means, WAT and SOM) performs worse than those supervised methods. Roughly, supervised methods can achieve a accuracy of over 90% that is twice higher than unsupervised methods. SOM performs the best in unsupervised method. Its accuracy is even a little bit higher than DT (supervised method).

Theoretically, RF can be seen as an improvement of DT, and CNN can be considered as an extension of BPNN. From the experiment results, the improved methods work much better.

## References


[1] A. Belaid and J.-P. Haton, "A syntactic approach for handwritten mathematical formula recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 1, pp. 105–111, 1984.

[2] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[3] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "Lerec: A nn/hmm hybrid for on-line handwriting recognition," *Neural Computation*, vol. 7, no. 6, pp. 1289–1303, November 1995.

[4] L. Jackel, M. Battista, H. Baird, J. Ben, J. Bromley, C. Burges, E. Cosatto, J. Denker, H. Graf, H. Katseff, Y. LeCun, C. Nohl, E. Sackinger, J. Shamilian, T. Shoemaker, C. Stenard, I. Strom, R. Ting, T. Wood, and Z. C., "Neural-net applications in character recognition and document analysis," in *Neural-Net Applications in Telecommunications*. Kluwer Academic Publishers, 1995.

[5] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," in *Neural Networks: The Statistical Mechanics Perspective*, J. H. Oh, C. Kwon, and S. Cho, Eds. World Scientific, 1995, pp. 261–276.

[6] Y. LeCun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of learning algorithms for handwritten digit recognition," in *International Conference on Artificial Neural Networks*, F. Fogelman and P. Gallinari, Eds. Paris: EC2 & Cie, 1995, pp. 53–60.

[7] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[8] Y. LeCun, "Pattern recognition and neural networks," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 1995.

[9] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, vol. 60, 1995.

[10] Wikipedia. (2014) Mnist database. [Online]. Available: http://en.wikipedia.org/wiki/MNIST_database#cite_note-1

[11] Y. LeCun, C. Cortes, and C. J. Burges. (2014) The mnist database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/index.html

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[13] D. Keysers, T. Deselaers, C. Gollan, and H. Ney, "Deformation models for image recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 8, pp. 1422–1435, 2007.

[14] B. Kgl and R. Busa-Fekete, "Boosting products of base classifiers," *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 497–504, 2009.

[15] D. Decoste and B. Schölkopf, "Training invariant support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 161–190, 2002.

[16] D. Claudiu Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *arXiv preprint arXiv:1003.0358*, 2010.

[17] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," in *Journal of Machine Learning Research*, 2004, pp. 1391–1415.

[18] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.

[19] Wikipedia. (2014) Random forest. [Online]. Available: http://en.wikipedia.org/wiki/Random_forest#cite_note-elemstatlearn-9

[20] LISALab. (2014) Convolutional neural networks (lenet)—deeplearning 0.1 documentation. [Online]. Available: http://deeplearning.net/tutorial/lenet.html

[21] Y. LeCun. (2013) Lenet-5, convolutional neural networks. [Online]. Available: http://yann.lecun.com/exdb/lenet/

[22] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012, supervised by Associate Professor Ole Winther, owi@imm.dtu.dk, DTU Informatics, and Morten Mørup, mm@imm.dtu.dk, DTU Informatics. [Online]. Available: http://www.imm.dtu.dk/English.aspx

[23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[24] Y. Huang and C. Suen, "The behavior-knowledge space method for combination of multiple classifiers," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1993, pp. 347–347.