Classification of Handwritten MNIST digits using Machine Learning Algorithms

Shrilekha Singh   Preksha Pansheriya   Venkatesh Tanniru   Nikhil Takappa Saunshi

***Abstract*—The MNIST database of handwritten digits, formed by Yann LeCun of NYU, has a total of 70,000 examples from approximately 250 writers. The images are 28*28 in size. There are 60,000 images in training test while 10,000 images in the test set. The project aims to implement several classification algorithms namely kNN, logistic regression, SVM, and ensemble learning for accurate handwritten digit recognition. For each of the algorithm, we will show how we find its optimal parameters, and test error. We concluded that SVM with rbf kernel is giving the minimum error rate among all classifier.**

***Index Terms*—MNIST digit, logistic regression,SVM, Ensemble learning, kNN**

I. Introduction and Motivation

Computer Vision has been a growing field intersecting many disciplines such as computer science, mathematics, and engineering. The interest in these type of problems has been increased over the years because of its potential application. In MNIST Digit Recognition we are building the classifiers based on training set and applying that classifiers to the test set to correctly predict the digit. We chose this problem as it is highly regarded as the steppingstone to the exciting field of computer vision. This paper outlines different Machine Learning Algorithm on MNIST data set. The MNIST database of handwritten digits, formed by Yann LeCun of NYU, has total of 70,000 examples from approximately 250 writers. The images are 28*28 in size. There are 60,000 images in training set while 10,000 images in the test set. The digits in both the dataset are labeled as any number between 0 to 9.

Since the MNIST dataset is quite popular in the research community. We were motivated to know what are the proven ways which we can use to handle this data, apply algorithms and infer the results. At present, there is a wide need for solving problems that involve high dimensional data. The chosen problem is on similar lines and throw opens the opportunity to learn about dimensionality reduction and how to create a visualization to get a sense of the data. This problem allowed us to apply ML algorithm that we learned in the class lectures. As the problem is a highly researched one, it allowed us to know if we have implemented our solution in the right way or not. The right execution solidifies our learning and motivates us to go further in the field of Computer Vision.

II. Related work

In the paper Comparison of classifier methods: A case study in handwritten digit recognition by Yann LeCun and others discusses about MNIST dataset in detail. They started with NIST (National Institute of Standard Institute) data set. They found less than 1% validation error but got very high test error. They concluded that NIST training data set contains handwritten digit of paid US census workers, while test set contains the handwritten digits of high school students. For this reason, they modified the data and divided the US census workers data into test and train. In this paper authors have applied classifiers such as baseline linear, baseline nearest neighbour, LeNet 1, LeNet 4, tangent distance classifier.Boosted net gave the best score.

Since, the data set in MNIST is not lineally separable and multiclass, the original optimal margin classifier algorithm developed by Vapnik, Boser, and Guyon gives accurate results only if the data is linearly separable in Z-space (transformed space). This technique was extended by Cortes and Vapnik. But they did not discuss in detail about how they deal with different classes. In the paper, A Comparison of Methods for Multi-class Support Vector Machines, author suggest that one-against-one and DAG methods are more suitable than one-against-all.

The paper Shape matching and object recognition using shape contexts discusses recognizing digits by measuring similarity between the shapes. Their approach involved three stage process. It involves solving correspondence problem between the two shapes and using to estimate an aligning transform. Last step is to compare the distance as the sum of matching errors between corresponding points. Their approach with 20000 training data set gave error rate of 0.63.

III. Data Preprocessing and Visualization

A. Data Vizualization using t-SNE

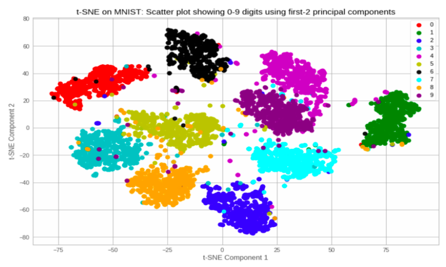


Fig. 1. Visualization of digits using t-SNE

To better visualize the high dimensional digit data set we used the algorithm, namely, t-SNE. It helps to visualize the high dimensional data in 2-D or 3-D. It can be used for dimension reduction as well but we are using it for data visualization in our paper.

t-SNE algorthim is mainly divided into two parts :

1) The first part of the algorithm consists of constructing a probability distribution between the high-dimensional data points in such a way that similar points have a high probability of being chosen for the embedding, while dissimilar points have a small probability of being picked. Let x_i and x_j are two points in high dimensional space. Then the conditional probability is given by using Gaussian distribution:

$$p_{i|j} = \frac{\frac{exp(-||x_i - x_j||^2}{2\sigma_i^2}}{\sum_k (1 + ||y_k - y_l||^2)^{-1}} \tag{1}$$

$p_ij = \frac{p_{i|j} + p_{j|i}}{2}$. The similarities between two points y_i and y_j. Here $p_j|i$ is a conditional probability that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under the Gaussian distribution centered at x_i. low-dimensional models of x_i and x_j are measured using a normalized heavy-tailed kernel points conditional property is given by using t-distribution. It is given by:

$$q_{i|j} = \frac{((1 + ||y_i - y_j||^2)^{-1}}{\sum_k (1 + ||y_i - y_j||^2)^{-1})} \tag{2}$$

2) Similar steps are performed in the low dimensional data and then we minimize the Kullback-Leibler divergence between the two joint probability distributions P (in high dimension) and Q (in low dimension) is given by:

$$C = KL(P||Q) = \sum_i \sum_j (p_{ij} \log \frac{p_{ij}}{q_{ij}}) \tag{3}$$

In our project, we applied t-SNE in python.Figure 2 shows the output from python in 2-D where all 10 labels can be visualized explicitly.

B. Digit deskewing

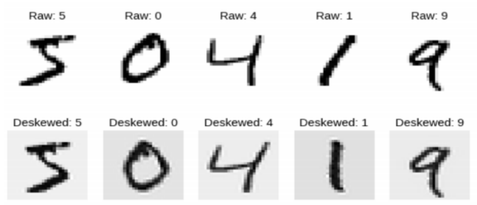


Fig. 2. The first row shows the numbers without any transformation. Second, after we de-skew the data

It is very natural that every person has different handwriting which poses several challenge to machine. One of the challenge is change in orientation is of each digit. This can led to misclassification of a digit. Therefore, we deskewd each digit to improve the classification. Figure 2 shows some examples of deskewed image comapared to raw image from our code.

C. Dimension Reduction using PCA

Dimension reduction maps the high dimensional data into a lower dimension. Data in low dimension retains the most of information and discard the features with very less variation within it. It helps to mitigate the computational cost as well as help to visualize the data. The MNIST data in our case has 784 dimensions, which is difficult to deal with.

In our project, we have used Principal Component Analysis for dimension reduction. It attempts to get rid of less informative dimensions to save the computational cost. PCA constructs a small number of linear features to summarize the input data. The idea is to rotate the axes so that the important dimensions in this new coordinate system become self evident and can be retained while the less important ones get discarded. But, it should be done carefully. Reducing large number of feature may result in loosing important information. So, the idea is to minimize the reconstruction error.

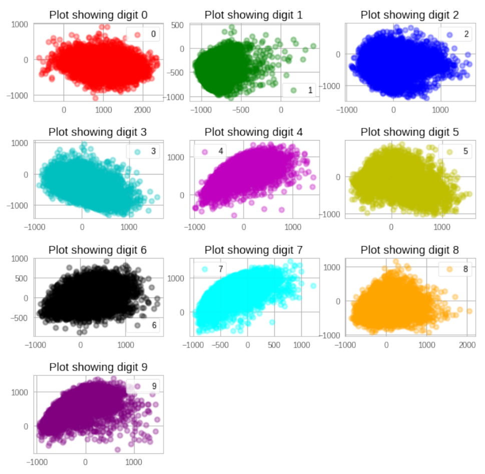


Fig. 3. Class-wise digit visualization using global PCA

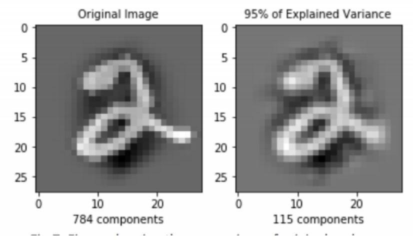


Fig. 4. Figure showing the comparison of original and reconstructed image

In PCA, dimensions with largest variation corresponds to one with eigen vector with largest largest eigen value. Consider a vector x (in the standard coordinate system) and some other coordinate system v_1,...,v_d, z_1, . . . , z_d can be defined as new coordinate system. Considering all components of new coordinate system:

$$x = \sum_{i=1}^{d} z_i v_i \tag{4}$$

Now consider the top k components of new coordinate system:

$$\hat{x} = \sum_{i=1}^{k} z_i v_i \tag{5}$$

Reconstruction error is defined as the difference between equation (4) and equation (5). It is captured as:

$$||x - \hat{x}||^2 = \sum_{i=k+1}^{d} (z_i)^2 \tag{6}$$

In order to have good coordinate system, this reconstruction error has to be as small as possible. In our project, after the centering the deskewd data we use the number of features corresponding to 95 percent of variance explained by the new coordinate system. It gives us 115 number of components. Figure 5 shows the relation between variance explained by the PCA and corresponding number of components. Figure 4 shows the comparison between image with 784 features and image with 115 features. We can see that our PCA does not loose important information and at the same time reduce the dimension to save the computation time. Figure 3 shows the class-wise global PCA in 2 dimensions.

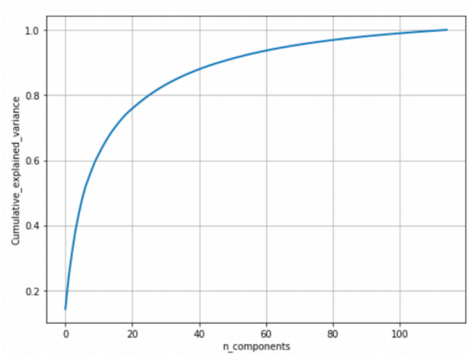


Fig. 5. Chart showing the cumulative-explained-variance gain on selecting the top-N principal components

IV. Classifiers

A. k-nearest neighbour

The k-nearest neighbour method works by assigning the label of closest data point in training set. Shortest distance is calculated by using euclidean distance formula. Let's x and y be any point in 2-D. Then euclidean distance is given by $\sqrt{(x-y)^T(x+y)}$. Here k represents number of neighbour to be considered to classify any new data point. For example, if k =1, it will consider only 1 neighbour to determine the final label of the new data point. If k is greater than 2, then will assign on the basis of majority.

In our project, to tune the parameter k, we applied K-fold cross validation method and chose the k which is giving the lowest validation error. From figure 6 and 7, best k is 3 for reduced dataset with 115 components and 46 components respectively. Figure also shows the test error curve which is

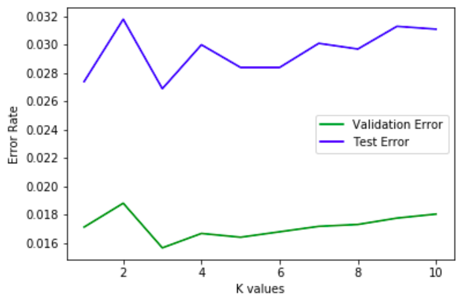


Fig. 6. performance of KNN on test and validation dataset with 115 dimension

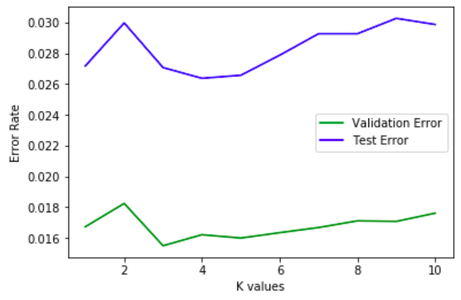
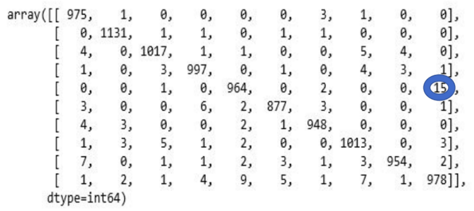


Fig. 7. performance of KNN on test and validation dataset with 46 dimension

also lowest at k =3. 3-NN performs really well with error rate of 1.45 % on reduced data set with 115 components and 1.51% with 46 components. Figure 8 shows the confusion matrix for dataset with reduced dimension (115 components). It clearly shows that 15 times 9 is misclassified as 4.



```
array([[ 975,    1,    0,    0,    0,    0,    3,    1,    0,    0],
       [   0, 1131,    1,    1,    0,    1,    1,    0,    0,    0],
       [   4,    0, 1017,    1,    1,    0,    0,    5,    4,    0],
       [   1,    0,    3,  997,    0,    1,    0,    4,    3,    1],
       [   0,    0,    1,    0,  964,    0,    2,    0,    0,   15],
       [   3,    0,    0,    6,    2,  877,    3,    0,    0,    1],
       [   4,    3,    0,    0,    2,    1,  948,    0,    0,    0],
       [   1,    3,    5,    1,    2,    0,    0, 1013,    0,    3],
       [   7,    0,    1,    1,    2,    3,    1,    3,  954,    2],
       [   1,    2,    1,    4,    9,    5,    1,    7,    1,  978]],
      dtype=int64)
```

Fig. 8. confusion matrix created with 115 dimension

B. Logistic Regression

From given dataset, it is used to predict binary outcome of dependent variable (yes/no, true/false) with other independent variables. Ordinal logistic regression deals with ordered dependent variables whereas multinomial logistic regression deals with data having three or more unordered outcomes possible. The sigmoid function is used to map predictions to probabilities as given below:

$$\sigma(z) = \frac{1}{1 + \exp^{-z}} \tag{7}$$

In our case, it is the extended case of classical logistic regression. Since we have more than two outcomes. The probability that the responses on observation i take on one of the m+1 possible outcomes can be modeled as

$$P(y_i) = \frac{\exp[x_i'\beta^m]}{1+\sum_{j=1}^{m}\exp[x_i'\beta^m]} \tag{8}$$

In our experiment, we applied logistic regression for one-vs-one, one-vs-all, and multinational scenario for data set with 115 and 46 dimensions.The histogram in figure 9 shows error rate vs time taken for execution is generated for three cases: one vs one,one vs rest and multinomial reduction having 95% and 85% variance respectively. It is observed that one vs rest classification has maximum error rate for 85$_{\%}$ variance compared to other two cases.

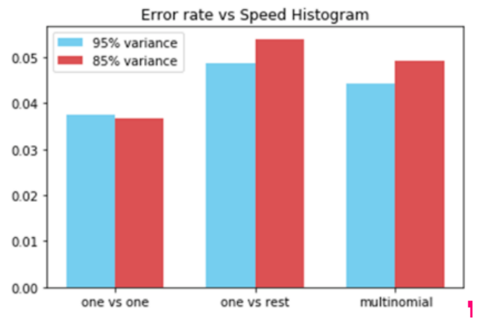


Fig. 9. Histogram comparing the test error in three cases: multinomial, one-vs-one, and one-vs-all classifier on MNIST data set with 115 and 46 components

C. Support Vector Machines

A data set is said to be linearly separable if there exists a line which can separate the one class of data with another class in 2 dimension (hyper-plane in multidimensional). The main idea of the Support Vector Machine is to give a decision boundary that maximizes the separation between two classes. But in our MNIST dataset, we have 10 labels. Therefore we used the concept of One-Vs-All and One-Vs-One to train the classifier. One-Vs-All considers one label as one class and combine remaining labels in one class. One-Vs-One considers one label vs another label. So, there are total $\frac{10!}{2!(10-2)!}$ ways i.e. 45.

1) Linear SVM: Since the data was not totally linearly separable, some mis-classification can be expected. Soft margin mathematical formation of SVM can be obtained by minimizing following objective function:

$$minimize\ \frac{1}{2}w^TW + C\sum_{n=1}^{N}\epsilon_n \tag{9}$$

$$subject\ to\ y_n(w^TX_n+b) \geq 1-\epsilon_n \tag{10}$$

where $\epsilon_n \geq 0$ for n = 1,2.....N C is called regularization parameter.In our experiment, we keep small value of C, which allowed SVM to have large margin. Linear SVM gives the error rate of 3.63 % with preprocessed and reduced data-set.

2) Kernel SVM: Since, our data set is not linearly separable, we used the kernel trick which transforms the data in z-space where data is linearly separable. In our project we used two kernels: polynomial with degree 3 and Gaussian kernel.

Polynomial kernel with degree 3 maps the data set in Z-space, where the data is linearly separable. We get 1.2 % error on reduced data set with One-Vs-One and 1.16 % with One-Vs-All.

Our study shows that Gaussian kernel gives the best performance. It is given by:

$$k(x_i,x_j) = \exp\frac{-||x_i-x_j||^2}{2\sigma^2} \tag{11}$$

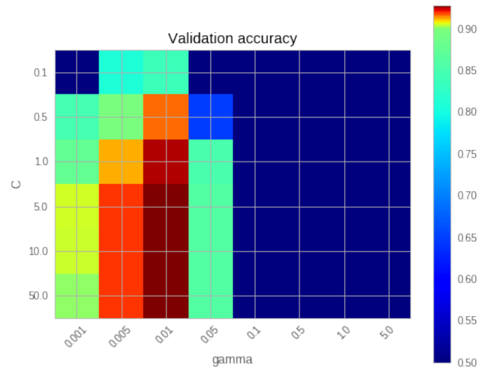


Fig. 10. Chart showing the score against each of the model parameters inside the grid

The challenge in Gaussian kernel is to get the optimal sigma. since, its the multiclass problem, optimizing the parameters is time taking. Due to time constrains We tuned the model parameter using grid search. Since, running Gaussian kernel is really computationally costly. We used this classifier with 20 percent of the training data set. Figure 10 shows the grid search output. Best parameter found are "C" = 5, gamma = 0.01. Gaussian kernel gives the the error rate of 2 % when trained when applied on 20 percent training data points.

D. Ensemble Learning

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In our project we are using 2 ensemble learning algorithms. They are Bagging (Bootstrap aggregating) and Random Forest.

1) Parameters: Parameters are either used to increase the predictive power of the model or to make it easier to train the model.

1) Max Features: It is the maximum number of features allowed in an individual tree. Since we have too many features, we found that the max feature of sqrt worked best for our project.
2) n_estimators: It is the number of trees to build before taking the average of predictions. In our project, we have tried with different values of estimators. We found that

the accuracy is directly proportional to the nestimators. Also, higher number of n_estimators made the code slower to run.

3) random_state: It is a parameter which makes the algorithm faster and easy to replicate. Same value of random_state will produce the same results when we pass the same data to the model.
4) oob_score: It is a major parameter which we used to train and validate the model. We can call this as a cross validation method of random forest. The oob_score is method is much faster and tags every observation used in different trees. And then it finds out a maximum vote score for every observation based on only trees which did not use this particular observation to train itself.

2) Bagging (Bootstrap aggregating: Bagging involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set.

In our project, we have trained the model with different number of parameters discussed above. Initially we ran the algorithm with different number of trees to find out the optimal number of trees. We ran the algorithm and found that 200 number of trees gives the optimal result considering the usage of resources. Then we ran the algorithm with 200 number of trees, random_state of 42 to come up with the accuracy, error and run time of the model. The same has been repeated with 85 % variance of data.

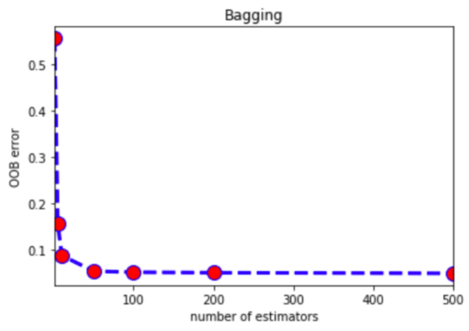


Fig. 11. error vs no. of estimators in bagging

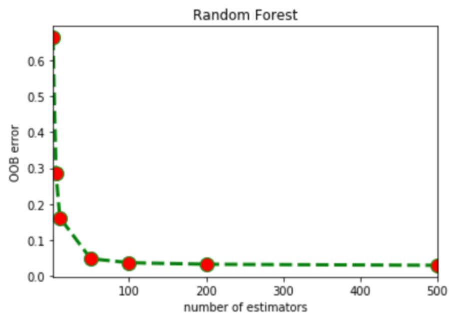


Fig. 12. error vs no. of estimators in random forest

3) Random forest: Random forests or random decision forests are operated by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. The main advantage of Random forests is, they correct for decision trees habit of overfitting to their training set.

As discussed above with bagging, the same procedure has been used for random forest algorithm to draw conclusions. In this algorithm, we added an extra parameter max_features of "sqrt" to get the best accuracy. That is, we ran the algorithm with 200 number of trees, random_state of 42 and max_features of "sqrt" to come up with the accuracy, error and run time of the model. The same has been repeated with 85% variance of data.

The algorithm with 95% variance of data took more time to run compared to 85% variance of data. The accuracy of the data is better with 85% variance of data compared to the 95% variance of data. So, from the observations,85% variance of data worked best for Bagging

V. Conclusion

Table 1 summarizes all the results which includes run time as well error rate for each classifier. Among all classifiers, SVM with polynomial degree 3 kernel gives the lowest error rate with 1.16 %. Compared to rbf and polynomial kernel, linear kernal gives poor performance. Since, SVM involves optimization, tuning parameter is computationally expensive. We found that among all classifier ensemble learning gave poor performance in terms of error. Since our experiment for each classifier is performed across several machines. So, its hard to come to common consensus to conclude if a algorithm has the lowest running time.

VI. Contribution

1) Shrilekha Singh: Data pre-processing, data vizualization, SVM and its results analysis
2) Preksha Pansheriya: Logistic regression and its result analysis
3) Nikhil Takappa Saunshi: Ensemble learning and its result analysis
4) Venkatesh Tanniru: kNN and its result analysis

References

[1] Y.LeCun, L. Bottou,C. Cortes,H.Drucker,I.Guyon,V. Vapnik," Comparison of Classifier Methods: A casestudy in handwritten digit recognition", In J. H. Oh, C. Kwon, S. Cho (Eds.), Neural networks: The statistical mechanics perspective (pp. 261-276). World Scientific.

[2] S.Belongie, J.Malik, J.Puzicha,"Shape Matching and Object Recognition Using Shape Contexts", IEEE Transactiosns on pattern analysis and machine intelligence, Vol. 24, N0. 24, april 2002

[3] C.Burges, B. Scholkopf, "Improving the Accuracy and Speed of Support Vector Machines",Denver, Colorado December 03 - 05, 1996

[4] C.Hsu, C. Lin,"A comparison of methods for multiclass support vector machines",IEEE Transactions on Neural Networks (Volume: 13 , Issue: 2 , Mar 2002)

[5] H. Zou, T. Hastie, and R. Tibshirani, Sparse Principal Component Analysis, 01 Jan 2012

[6] Y.LeCunn, L. Bottou, Y. Bengio,"Gradient Based Learning Applied to Document Recognition",1998

[7] L. Maaten, G. Hinton,"Visualizing Data using t-SNE",9(Nov):2579–2605, 2008.

TABLE I
COMPARISON OF ALL MODELS

Model detail	***Preprocessing***	***error rate***(%	***run time (sec)***
Logistic regression (one vs one)	deskewing and PCA (115 components)	3.74%	102
Logistic regression (one vs one)	deskewing and PCA (46 components)	3.61	31.6
Logistic regression (one vs all)	deskewing and PCA (115 components)	4.86	165
Logistic regression (one vs all)	deskewing and PCA (46 components)	5.4	38
Logistic regression (multinomial)	deskewing and PCA (115 components)	4.43	446
Logistic regression (multinomial)	deskewing and PCA (46 components)	4.93	189
kNN	deskewing and PCA (115 components)	1.45	130
kNN	deskewing and PCA (46 components)	1.51	48.6
Linear SVM	deskewing and PCA (115 components)	3.63	189
3 degree polynomial SVM (one vs one)	deskewing and PCA (115 components)	1.2	157
3 degree polynomial SVM (one vs rest)	deskewing and PCA (46 components)	1.16	119
3 degree polynomial SVM (one vs all)	deskewing and PCA (115 components)	1.2	678.48
3 degree polynomial SVM (one vs rest)	deskewing and PCA (46 components)	1.36	671
SVM (Gaussian kernal)	raw data	2	1092
Bagging	deskewing and PCA (115 components)	5.06	1898.47
Bagging	deskewing and PCA (46 components)	4.62	712.07
Random Forest	deskewing and PCA (115 components)	3.28	138.83
Random Forest	deskewing and PCA (46 components)	2.96	85.26