Scientific Research Publishing

Journal of Intelligent Learning Systems and Applications, 2017, 9, 55-68
http://www.scirp.org/journal/jilsa
ISSN Online: 2150-8410
ISSN Print: 2150-8402

A Minimal Subset of Features Using Feature Selection for Handwritten Digit Recognition

Areej Alsaafin, Ashraf Elnagar

ML & ALP Research Group, Department of Computer Science, University of Sharjah, UoS, Sharjah, UAE
Email: ashraf@sharjah.ac.ae

How to cite this paper: Alsaafin, A. and Elnagar, A. (2017) A Minimal Subset of Features Using Feature Selection for Handwritten Digit Recognition. *Journal of Intelligent Learning Systems and Applications*, **9**, 55-68.
https://doi.org/10.4236/jilsa.2017.94006

Received: June 1, 2017
Accepted: October 28, 2017
Published: October 31, 2017

Abstract

Many systems of handwritten digit recognition built using the complete set of features in order to enhance the accuracy. However, these systems lagged in terms of time and memory. These two issues are very critical issues especially for real time applications. Therefore, using Feature Selection (FS) with suitable machine learning technique for digit recognition contributes to facilitate solving the issues of time and memory by minimizing the number of features used to train the model. This paper examines various FS methods with several classification techniques using MNIST dataset. In addition, models of different algorithms (*i.e.* linear, non-linear, ensemble, and deep learning) are implemented and compared in order to study their suitability for digit recognition. The objective of this study is to identify a subset of relevant features that provides at least the same accuracy as the complete set of features in addition to reducing the required time, computational complexity, and required storage for digit recognition. The experimental results proved that 60% of the complete set of features reduces the training time up to third of the required time using the complete set of features. Moreover, the classifiers trained using the proposed subset achieve the same accuracy as the classifiers trained using the complete set of features.

Keywords

Digit Recognition, Real Time, Feature Selection, Machine Learning, Classification, MNIST

1. Introduction

Handwriting recognition is the ability of recognizing handwritten text from a scanned file, image, touch-screen or other tools and converting it into an editable text [1]. Handwriting recognition is a quite complex problem. The ideal goal of designing a handwriting recognition system with 100% accuracy is not a possible thing that can be achieved. This is because even humans do not have the ability of recognizing any handwritten text without any doubt [2]. There are various reasons that may cause inaccuracy such as pattern twist, reality of unwanted objects, confused patterns, and dwindling of paper. Many different initiatives have been proposed, however, the results are still not perfect [3]. Even now there is no approach that solves the problem completely and efficiently [4]. The recognition of handwritten digits is not a new problem. It is a problem that has continued to be an active topic in research for several reasons [5]:

- The problem is suitable for image processing and pattern recognition using a small number of classes.
- Availability of standard benchmark datasets reduces the effort and time spent in preprocessing the data.
- A lot of work and research have been done in this field that can be cited and built on.
- The practical applications encourage more research to be done.
- Improvements in classification accuracy using existing approaches continue to be achieved using new approaches.

In addition, handwritten digit recognition is an active subject in OCR (Optical Character Recognition) applications and pattern classification/learning research [6]. OCR applications like postal mail sorting, bank check processing, and form data entry require high accuracy and speed techniques to achieve a satisfactory performance. The aforementioned reasons motivate us to proceed with this study.

In this paper, we employ a Feature Selection (FS) method in order to select a subset of relevant features using the MNIST dataset. The FS method used in this paper is called Feature Importance. This paper also implements various classification techniques in order to study their suitability for digit recognition. It presents a statistical analysis that compares models of different algorithms, which are:

- Linear classification: Multinomial Logistic Regression (MLR);
- Non-linear classification: Decision Tree (DT);
- Ensemble classification: Random Forest (RF) and Boosted Trees (BT);
- Deep learning: Convolutional Neural Network (CNN).

The goal behind using classifiers of different algorithms is to find out the most suitable and efficient algorithm for digit recognition in terms of training time, accuracy, and confusion matrix. In addition, the purpose of introducing this comparison is to limit the algorithms that can be used to solve such a problem and devote more effort to develop the suitable techniques by doing more experiments and research.

The rest of this paper is organized as follows. Section 2 reviews related literature. Sections 3 and 4 include description of the dataset, FS, and classification techniques used in this paper. The stages of the performed experiment and the obtained results are presented in Section 5. Section 6 includes some discussion before concluding this paper in Section 7.

2. Related Work

Much related work in the literature focuses on digit recognition using different machine learning techniques. There are many studies focus on combining and developing various machine learning techniques in order to improve the performance of digit recognition in terms of accuracy. However, there are some limitations in these studies. First, very few studies have focused on FS methods that can improve the digit recognition performance. Although, FS methods are very important to enhance the time required to train a model to add more efficiency to applications especially for online applications. Most of the studies that use FS have used complex methods that require some extra time to select the most relevant features in a given dataset. Many of them tend to rely only on a specific FS method such as principal component analysis (PCA). Most of these studies have used the MNIST dataset but each study uses different features without presenting any information about the selected features or the number of features selected by a specific FS method. Therefore, even now there is no exact answer to the question about which features of the MNIST dataset are more relevant/representative to be used for digit recognition. Second, the time required to train the classifiers/models is not considered in the evaluation stage in many related studies. The time factor is crucial in online applications. However, a lot of work has been reported about combining various classification techniques that have been evaluated in terms of the accuracy and detection rate without taking into account the training or the testing time.

From another perspective, the experiments of handwritten digit recognition available in the literature differ in several factors like sample data, feature representation, pre-processing techniques, learning algorithms, and classifier structure. There are only few works that compare classification techniques using the same dataset and feature subset/set. Among pattern recognition problems, digit recognition has been widely addressed for its usability and ease of implementation. Some of the most popular datasets are NIST Special Database 19 (SD19), Modified National Institute of Standards and Technology (MNIST), etc. The NIST SD19 contains a massive number of handprinted forms and character images, thus researchers usually use different subsets of the dataset for the purpose of training and testing. In contrast, the MNIST dataset is divided into standard training and testing sets. Therefore, we collected some work reported on the MNIST dataset.

LeCun *et al.* [7] reported different accuracies obtained from various classifiers. The highest accuracy, 99.30%, was obtained via boosted convolutional neural network (CNN) that is trained using distorted data. Simard *et al.* [8] enhanced distorted sample generation as well as the implementation of CNN. This led to a small improvement in test accuracy, 99.60%. Liu *et al.* [6] used gradient direction features to implement several classification methods. The test accuracies that have been achieved are 99.42% by polynomial classifier, 99.58% by SVM classifier, and over 99% by many other classifiers. Holmstrom *et al.* [9] compared different statistical and neural classifiers based on PCA features. However, the PCA feature does not perform satisfactorily. Liu *et al.* [10] claim that training classifiers using MNIST dataset without using feature extraction methods shows inferior performance. This motivates us to compare the performance of various classification models trained with a subset of features against the complete set of features available in the MNIST dataset.

3. Dataset & Feature Selection

1) MNIST dataset

In this paper, we use MNIST dataset. The MNIST is a dataset developed by LeCun, Cortes and Burges for evaluating machine learning models on the handwritten digit classification problem [11]. It has been widely used in research and to design novel handwritten digit recognition systems. The MNIST dataset contains 60,000 training cases and 10,000 test cases of handwritten digits (0 to 9). Each digit is normalized and centered in a gray-scale (0 - 255) image with size 28 × 28. Each image consists of 784 pixels that represent the features of the digits. Some examples from the MNIST dataset are shown in **Figure 1**. The MNIST dataset is balanced over the ten classes (0 - 9). **Figure 2** shows the percentage of each class in the MNIST dataset.

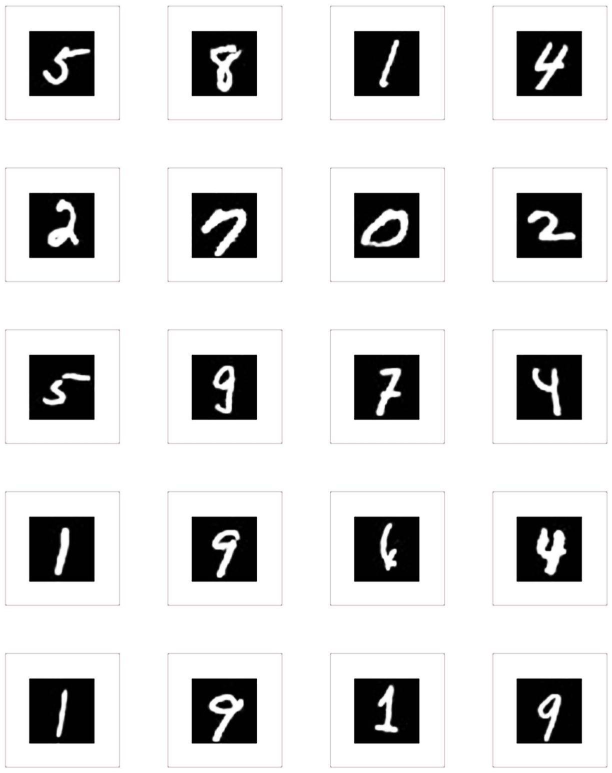
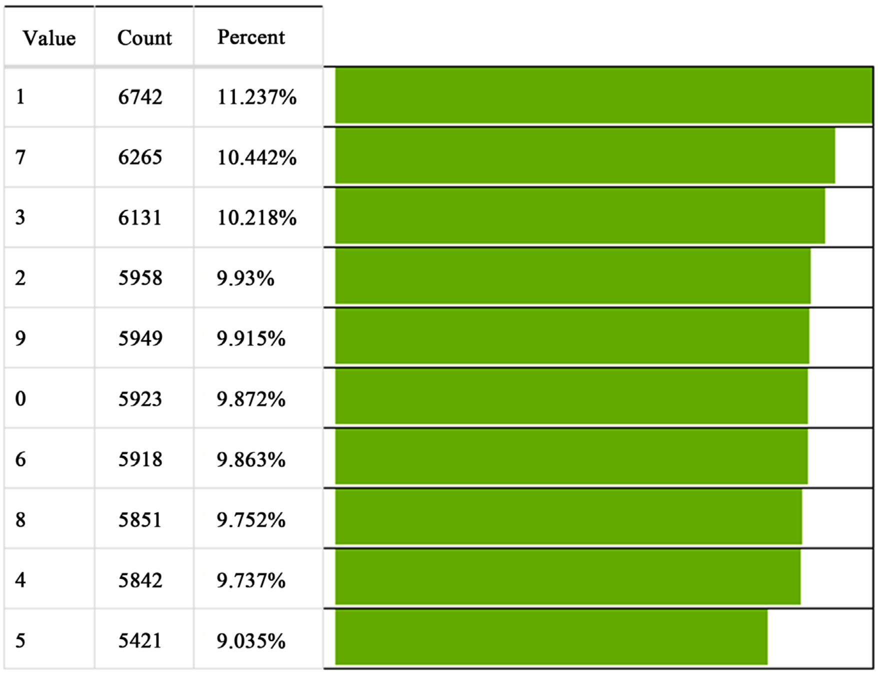


Figure 1. Examples of MNIST dataset.



Value	Count	Percent
1	6742	11.237%
7	6265	10.442%
3	6131	10.218%
2	5958	9.93%
9	5949	9.915%
0	5923	9.872%
6	5918	9.863%
8	5851	9.752%
4	5842	9.737%
5	5421	9.035%

Figure 2. Class percentages in MNIST dataset.

2) Feature selection methods

Features that are used in any machine learning method have a huge influence on the obtained results. Having Irrelevant or partially relevant features can negatively affect performance of many models, especially linear algorithms like logistic regression [12].

Feature selection (FS) is a process of selecting features that contribute most to the prediction variable. An FS method selects a subset of relevant features and discards irrelevant/redundant features. The benefits of performing FS before data modeling are [12]:

- Reduces Overfitting: Less redundant data implies less chance to make decisions based on noise.
- Improves Accuracy: Less misleading data implies modeling accuracy improves.
- Reduces Training Time: Less data implies that algorithms train faster.

In general, there are two types of FS methods: filter method and wrapper method. The filter method selects a subset of the most relevant features based on the characteristics of the dataset. This type of FS methods is independent of the classification algorithms unlike the wrapper methods that use a predetermined classifier in order to evaluate the selected subset of features. The wrapper method is more computationally expensive than the filter method [13]. In this research, we selected a wrapper FS method called Feature Importance. The Feature Importance is a method that estimates the importance of features using bagged decision trees like random forest and extra trees [12].

Our goal in this project is not to improve machine leaning algorithms but compare the performance of different algorithms using a subset of features against the complete set of features. Therefore, we use a simple FS method that serves our goals.

4. Classification Techniques

Classification is a process of finding a model/function that recognizes, describes, and differentiates two or more classes. The purpose of using a classification model is to predict the class of an object where its class label is unknown. Classification has a predictive nature—for a given set of features, the goal is to attempt to predict the value of another feature [14]. A classifier model is used to classify the actual data into defined classes. Ultimately, patterns need to exist in the data that can be exploited. The classification techniques used in this paper are explained below.

1) Multinomial Logistic Regression:

A Logistic Regression is a binary classifier that can distinguish between two classes. Multinomial Logistic Regression (MLR) is a logistic regression that is designed to solve multiclass/multinomial classification problems. In other words, Multinomial Logistic Regression is a model that predicts the probabilities of different possible outcomes of a categorically distributed dependent variable, given a set of independent variables [15]. It is used when the dependent variable is nominal and falls into one of many (*i.e.* three or more) classes/categories (e.g. the MNIST dataset has nine classes).

MLR classifier is commonly used as an alternative to naive Bayes classifier since it does not assume statistical independence of the random features that used as predictors. The MLR classifier is simple and requires a little time to learn, however, it becomes slow when use a very large number of classes to learn.

2) Decision Tree:

A Decision Tree (DT) classifier is a flow-chart-like structure, where the top-most node represents the root node of the tree, each intermediate node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf node indicates a class label. The paths from root to leaf represent classification rules.

DT is a simple model to understand and interpret. However, computation can get very complex if many outcomes are liked (*i.e.* a decision tree reaches a great depth). The DT classifier becomes biased in favor of attributes with more levels/depth; however, it can be easily combined with other decision techniques in order to make better and accurate decisions [16].

3) Random Forest:

A Random Forest (RF) classifier [17] is one of the most effective techniques that is used for predictive analytics. It is an ensemble learning classifier that combines decisions from a sequence of base classifiers in order to make a decision. RF is correct for decision trees' habit of overfitting to their training set [18]. Decision Trees that have great depth tend to overfit their training sets thus have low bias but very high variance. RF classifier reduces the variance by averaging multiple deep decision trees that have trained using different parts of the same training set. Although this process leads to a small increase in the bias, it improves the overall performance of the final model.

4) Boosted Trees:

A Boosted Trees (BT) or Gradient Boosting Machine (GBM) is an additive technique that is developed by Friedman [19]. It combines predictions from a sequence of base classifiers in order to make a prediction. Typically it is constructed of a number of simple/weak DT models where the output is the sum of the decisions of these base learners. The purpose of BT method is to achieve better predictive performance by minimizing the loss function when adding trees. Generally adding more trees makes the model more resistance to overfitting. Therefore, keep adding trees until no further improvement is observed can lead to build an efficient model that gives very accurate predictions.

5) Convolutional Neural Network:

A Convolutional Neural Network (CNN) is a deep learning method that can be used for the purpose of feature extraction as well as classification. Hence, CNN acts as a feed-forward network that extracts topological properties from images [20]. It extracts features from raw images (*i.e.* contain the intended pattern) in its first layers, and then classifies the pattern with its last layers [20].

CNN is a powerful technique for image processing as well as natural language processing. The main advantage of CNN is the high accuracy in its results. However, it requires high computational cost. In addition, it needs a lot of data to be trained. The complexity of the CNN slows down the training process thus it is necessary to use a good GPU to overcome this problem.

5. Experimental Results

As previously mentioned, we use the MNIST dataset to perform the experiment of this study. In addition, we use a FS method to be examined with different classifiers: MLR, DT, RF, and BT. At the last stage of this study, we compare the same classifiers used at the previous stage in addition to CNN. In this section, we describe the tools that used throughout the experiment of this study, describe all the experiment stages, and finally present and discuss the obtained results.

1) Hardware and Software Tools

Since we consider some factors (e.g. training time) that are affected by the tools used in the experiment, it is worth mentioning the hardware and software that have been used throughout this research. For the hardware tools, we used a laptop of MacBook Pro, 2.8 GHz Intel Core i7 with 4 GB RAM. All the setup and computations of this research have been performed on top of the CPU. For the software tools, GraphLab Create[1] is used for the implementation of this paper. GraphLab Create is a machine learning framework that is python-based libraries. It is designed to handle the major properties of real world data: scalable, variety, and complexity. The biggest advantage of using GraphLab is that it supports scalability, which allows using large datasets. It also supports different data

[1]https://turi.com/.

source such as JSON, CSV, HDFS/S3 and many more. Graphlab is a complete framework that has rich libraries for data transformation and manipulation. It is worth mentioning that we import some libraries from scikit-learn for the purpose of implementing the FS method used in this paper.

2) Pre-processing

Since the MNIST dataset is divided into standard training and testing sets, 80% and 20% respectively, we divide the MNIST training set into two proportions, 80% as a training set and 20% as a validation set. Therefore, the MNIST dataset is divided into three sets: training set (60%), validation set (20%), and testing set (20%). The training set is used to train the classifiers for the purpose of recognizing handwritten digits while the test set is used to assess the trained classifiers after fine-tuning them using the validation set.

Before training the classifiers, we select subsets of various sizes (*i.e.* number of features) using Feature Importance method. Table 1 presents the time required for selecting different subsets using Feature Importance method. Over all the subset sizes, the time required for selecting a subset is reasonable. The required time does not change much when selecting small or large subset (e.g. the required time for selecting subsets of 20% and 80% of features are 37.76 sec and 38.87 sec, respectively). However, few seconds are added to the required time whenever the number of features increases.

3) Study I: subset vs. complete set

At this study, we perform FS on the MNIST dataset in order to select the best subset of features to be compared with the complete set of features.

a) Select the best subset of features

In order to find the best subset, we train the MLR, DT, RF, and BT classifiers using all the subsets selected by the Feature Importance method at the pre-processing stage. The results show that the validation error does not change much after the subset of 60% of features. Moreover, the validation error becomes fairly stable by using subsets of more than 60% of features. Accordingly, the subset of 60% of features is used for the comparison with the complete set of features.

b) Train and Fine-tune classifiers

After we examined different sizes of features, we train the classifiers (MLR, DT, RF, and BT) using the subset of 60% of features as well as complete set of features (784 features).

The last step before we draw the comparison is to fine-tune the classifiers using the validation set in order to improve their performances. The classifiers trained with the subset of features as well as complete set of features are fine-tuned

Table 1. Time required to select various sizes of subset using feature importance method.

FS method	Time (sec) over subsets of various number of features				*Average*
	20%	**40%**	**60%**	**80%**	
Feature Importance	37.76	38.12	**38.44**	38.87	38.30

using the validation set. The MLR and the RF classifiers have been fine-tuned by increasing the number of iterations while the DT and BT classifiers have been fine-tuned by controlling the tree depth and the number of trees, respectively. As shown in Figure 3, the MLR and RF have the same validation error after 15 iterations. For the comparison, models with 50 iterations are enough since the validation accuracy stabilized in both classifiers. However, the RF has not really improved by increasing the number of iterations. The validation accuracy of the DT classifier stabilizes after depth 30 thus we limit the depth of the DT classifier to 30. Since the validation accuracy of the BT does not improve significantly after using 130 trees, we limit the number of trees to 130.

c) Compare subset against complete set

The last step of this study is to compare the performance of the classifiers using the subset of features (60%) against the complete set of features. As shown in Table 2 and Table 3, training the classifiers using a subset of features improves the training time as well as gives about the same test accuracy as the classifiers trained with the complete set of features. Moreover, the training time of all the classifiers has significantly reduced by using only 60% of the complete set of features.

4) Study II: suitable classification technique

In this study, we examine various classification techniques in order to find the most suitable techniques for handwritten digit recognition. The classification techniques used in this study are MLR, DT, RF, BT and CNN.

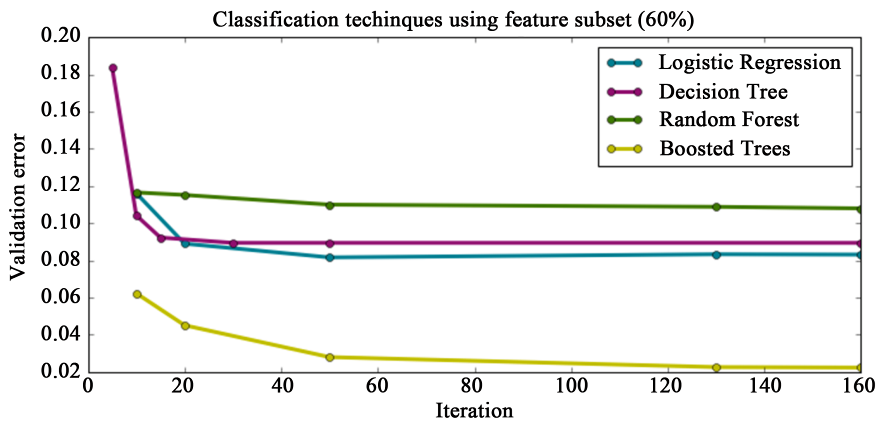


Figure 3. Validation error of classification techniques over different number of iterations.

Table 2. Evaluation of different classification techniques using subset of features (60%).

Metric	Classification Technique			
	MLR	*DT*	*RF*	*BT*
Training Accuracy	93.40%	98.97%	91.19%	100.00%
Validation Accuracy	91.84%	91.06%	89.00%	97.74%
Test Accuracy	92.42%	90.87%	90.08%	97.84%
Training Time (*sec*)	59.97	23.34	269.01	852.50

Table 3. Evaluation of different classification techniques using complete set of features.

Metric	Classification Technique				
	MLR	*DT*	*RF*	*BT*	*CNN*
Training Accuracy	93.80%	98.96%	91.05%	100%	97.64%
Validation Accuracy	91.88%	91.02%	88.92%	97.68%	98.37%
Test Accuracy	92.61%	90.94%	89.85%	97.91%	98.36%
Training Time (*sec*)	412.29	71.11	600.94	1396.03	780.18

a) Train and fine-tune the classification techniques

In order to have a fair and valuable comparison, all the classification techniques are trained using the complete set of features. The MLR, DT, RF, and BT classifiers are trained as same as they trained in the Study I. A one layer CNN has been trained to classify an input to one of the ten classes (0 - 9) available in the dataset. Similar to the other classifiers, the CNN is fine-tuned by increasing the number of iterations in order to fulfill the training criteria.

b) Compare the classification techniques

Table 3 presents the training accuracy, validation accuracy, test accuracy, and training time for all techniques after fine-tuning them based on the validation set. In this comparison, we did not consider the time required for testing since we use single classifiers (*i.e.* not combined classifiers) that do not require more than few seconds to predict any new input. As shown in Table 3, the MLR, DT, and RF have low test accuracy compared with the BT and CNN. However, they outperform the BT and CNN in terms of training time. The DT has the lowest training time (71.11 sec), although it has the lowest test accuracy (90.94%) among the four techniques. The CNN outperforms all the techniques in terms of test accuracy (98.36%). The BT has the highest training accuracy (100%) and the second highest test accuracy (97.91%). However, the BT requires the longest time to be trained among all other techniques (1396.03 sec).

Figure 4 shows the confusion matrix of the BT and the CNN since they have the highest test accuracy compared with other techniques. The confusion matrix shows a more detailed breakdown of correct and incorrect classifications of each class on the MNIST test set. It composed of target (actual) label, predicted label, and the count of predicting the target label as the predicted label. As shown in Figure 4, the BT and CNN has misclassified digit "5" 19 times, mostly it was classified as digit "3". The CNN outperforms the BT in classifying all the digits except digit "6" and "8". The highest misclassified digit using the BT classifier is digit "7" that has been classified as digit "2" 15 times, while digit "6" is the most misclassified digit using the CNN, 16 mistakes. Using the CNN, digit "6" was often classified as digit "0". However, the CNN was the best in classifying digit "0", only 3 mistakes. On the other hand, the BT classifier was the best in classifying digit "1", 9 mistakes. In general, we suggest that most of the mistakes made by the BT and CNN were due the same shape that some digits have (e.g. "0" and "6"). Figure 5 shows the common and individual mistakes of the CNN and BT when classify digit "7" as digit "2".

		Predicted Label									
		0	1	2	3	4	5	6	7	8	9
Target Label	0	970	0	1	0	0	1	3	1	3	1
	1	0	1126	1	3	0	1	1	1	2	0
	2	6	0	1007	5	1	0	0	7	5	1
	3	0	0	4	989	0	5	0	7	3	3
	4	1	0	4	1	961	0	3	0	4	8
	5	3	0	1	6	0	873	4	3	1	1
	6	5	2	0	0	2	4	941	0	4	0
	7	2	2	15	4	1	0	0	995	1	8
	8	3	1	1	2	2	2	4	1	953	5
	9	5	5	3	5	9	0	0	5	1	976

(a)

		Predicted Label									
		0	1	2	3	4	5	6	7	8	9
Target Label	0	977	0	0	0	0	0	0	1	2	0
	1	0	1129	0	2	0	0	1	2	0	1
	2	5	3	1009	3	0	0	0	9	3	0
	3	1	0	2	1001	0	2	0	1	2	1
	4	1	0	0	0	968	0	0	2	1	10
	5	2	0	0	8	0	873	3	2	2	2
	6	12	2	0	0	2	4	937	0	1	0
	7	0	2	6	0	1	0	0	1016	2	1
	8	2	0	2	6	1	5	2	6	945	5
	9	3	2	0	6	6	1	0	7	3	981

(b)

Figure 4. Confusion matrix: (a) BT confusion matrix; (b) CNN confusion matrix.

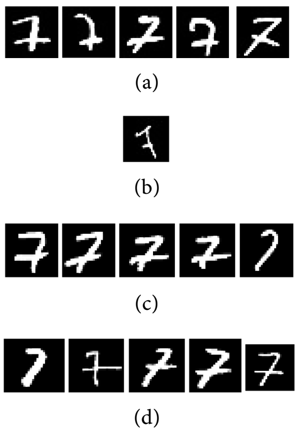


Figure 5. Digit "7" classified as digit "2" by CNN and BT (a) common mistakes between CNN and BT (b) individual mistakes by CNN (c) individual mistakes by BT.

6. Discussion

It is worth mentioning that the results obtained in the previous section can be optimized using different methods. We convinced of these results since our target was to find the most suitable classification techniques that can be used to solve the problem of recognizing handwritten digits. From this experiment, we can conclude that the CNN and the BT are the most suitable classification techniques among all other techniques used in this research or similar to them. Since the BT classifier requires a long time to be trained using the complete set of features, we suggest training the BT using a subset of 60% of features using FS method. This reduces the required time, complexity of the trained model, and required storage for digit recognition while still provides very similar performance to the model trained with the complete set of features.

As we discussed in the related work section, Liu *et al.* in [10] claim that training classifiers using the complete set of features available in the MNIST dataset gives unsatisfactory performance. However, the obtained results in the implementation section (Study I) show that the models trained with the complete set of features have accuracies not less than the accuracies obtained for the models trained using FS. Therefore, there is about 40% of features exist in the MNIST dataset are irrelevant/redundant features.

It is worthless to compare our results to other results since most of the related work has some limitations. Most of the related work available in the literature does not specify the type of the reported accuracy: training, validation, or testing accuracy. However, it is very important to identify the type of the obtained accuracy since the training accuracy could reach 100% if we kept fine-tuning the model until perfectly fit the training dataset. Although this leads to overfitting problem that causes a reduction in the test accuracy. We can notice from **Table 3** that the BT classifier achieves 100% training accuracy without affecting the test accuracy. This is because the BT is an ensemble classifier that tends to be robust to overfitting, however, it will eventually overfit.

Many studies fine-tune and assess models using the same data (testing set). This leads the test results to be overly optimistic, which means that the model is trained to give a good estimation on the test results. Therefore, assessing the models using the test set will be worthless since the model will give bad results once used to predict new data. To avoid this problem, the model should be fine-tuned using the validation set and then assessed using the test set.

The last point of this discussion is that some studies that use un-standard datasets do not present the distribution of the instances/examples available in the dataset. Imbalance datasets lead to have a high test accuracy with many false positive predictions. Therefore, the accuracy metric is not enough to evaluate the classification techniques since it does not capture everything. The confusion matrix is a very useful metric that shows the details of the predictions made by the trained models.

7. Conclusion & Future Work

This paper proposed a subset of 60% of features to train classification techniques for digit recognition. In addition, it has implemented and compared models of different algorithms: linear, non-linear, ensemble, and deep learning to study their suitability for digit recognition. The obtained results show that the proposed subset of features is not only minimized but also reduces the required time for training the model. Moreover, it reduces the computational complexity and lowers the required storage for digit recognition. The BT and CNN proved their suitability for digit recognition in terms of accuracy. However, the BT classifier requires using a subset of features since it needs a long time to be trained. We evaluated the performance in terms of accuracy, training time, and confusion matrix.

In future, we plan to examine more FS methods with different classification techniques. Combining various FS methods is another area to be explored. Based on our observations from the confusion matrices of the BT and CNN, some digits are well recognized by some classifiers while misclassified by other classifiers. This observation motivates us to try FS with a combined machine learning techniques in order to enhance the required time as well as accuracy.

References

[1] Lee, H., *et al.* (2012) Machine Learning Techniques in Handwriting Recognition: Problems and Solutions. In: Kulkarni, S., Ed., *Machine Learning Algorithms for Problem Solving in Computational Applications: Intelligent Techniques*, IGI Global, University of Ballarat, Australia, 12-29. https://doi.org/10.4018/978-1-4666-1833-6.ch002

[2] Sonka, M., Hlavac, V. and Boyle, R. (2014) Image Processing, Analysis, and Machine Vision. Cengage Learning, Stamford, USA.

[3] Jain, V., *et al.* (2016) Comparative Analysis of Machine Learning Algorithms in OCR. 3*rd International Conference on Computing for Sustainable Global Development* (*INDIACom*), New Delhi, India, 16 March 2016, 1089-1092.

[4] Kacalak, W., Stuart, K.D. and Majewski, M. (2007) Selected Problems of Intelligent Handwriting Recognition. In: Melin, P., Castillo, O., Ramírez, E.G., Kacprzyk, J. and Pedrycz, W., Eds., *Analysis and Design of Intelligent Systems Using Soft Computing Techniques*, Springer, Berlin Heidelberg, 298-305. https://doi.org/10.1007/978-3-540-72432-2_30

[5] Keysers, D. (2007) Comparison and Combination of State-of-the-Art Techniques for Handwritten Character Recognition: Topping the Mnist Benchmark. arXiv preprint arXiv:0710.2231.

[6] Liu, C.L., *et al.* (2003) Handwritten Digit Recognition: Benchmarking of State-of-the-Art Techniques. *Pattern Recognition*, **36**, 2271-2285. https://doi.org/10.1016/S0031-3203(03)00085-2

[7] LeCun, Y., Cortes, C. and Burges, C.J.C. (1998) The MNIST Database of Handwritten Digits. New York, USA. http://yann.lecun.com/exdb/mnist/

[8] Simard, P.Y., Steinkraus, D. and Platt, J.C. (2003) Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. *ICDAR*, **3**, 958-962.

[9] Holmstrom, L., *et al.* (1997) Neural and Statistical Classifiers-Taxonomy and Two

Case Studies. *IEEE Transactions on Neural Networks*, **8**, 5-17. https://doi.org/10.1109/72.554187

[10] Liu, C.L. and Fujisawa, H. (2008) Classification and Learning Methods for Character Recognition: Advances and Remaining Problems. In: Marinai, S. and Fujisawa, H., Eds., *Machine Learning in Document Analysis and Recognition*, Springer, Berlin Heidelberg, 139-161. https://doi.org/10.1007/978-3-540-76280-5_6

[11] Le Cun, Y., *et al.* (1998) Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, **86**, 2278-2324.

[12] Brownlee, J. (2016) Feature Selection for Machine Learning in Python. http://machinelearningmastery.com/feature-selection-machine-learning-python/

[13] Nnamoko, N., *et al.* (2014) Evaluation of Filter and Wrapper Methods for Feature Selection in Supervised Machine Learning. *Age*, **21**, 33-2.

[14] Han, J. and Kamber, M. (2000) Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems.

[15] Engel, J. (1988) Polytomous Logistic Regression. *Statistica Neerlandica*, **42**, 233-252. https://doi.org/10.1111/j.1467-9574.1988.tb01238.x

[16] Quinlan, J.R. (1987) Simplifying Decision Trees. *International Journal of Man-Machine Studies*, **27**, 221-234.

[17] Breiman, L. (2001) Random Forests. *Machine Learning*, **45**, 5-32. https://doi.org/10.1023/A:1010933404324

[18] Friedman, J., Hastie, T. and Tibshirani, R. (2001) The Elements of Statistical Learning. Vol. 1, Springer Series in Statistics, Springer, Berlin.

[19] Friedman, J.H. (2001) Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, **29**, 1189-1232. https://doi.org/10.1214/aos/1013203451

[20] Lauer, F., Suen, C.Y. and Bloch, G. (2007) A Trainable Feature Extractor for Handwritten Digit Recognition. *Pattern Recognition*, **40**, 1816-1824.