COMP5318 Machine Learning And Data Mining Assignment 2

Tutors: Fangzhou Shi | Canh Dinh

Group members: Jiaxuan Guo 460108267 | Yili Wang 470106260 | Kefu Duan 470306240

Abstract

Machine learning techniques have been used in many industrial fields. This report involves using machine learning techniques on one of the most popular computer version task, image classification. This article mainly discusses three different classifiers algorithms that are support vector machine (SVM), random forest (RF) and Convolutional Neural Network (CNN). We implement these three classifier algorithms and compare their performance and results based on the Fashion-MNIST dataset. The stages involving in the project include pre-processing dataset, classifier theory, classifier implementation details, and model performance. The performance and result evaluation mostly is examined by running time, accuracy and confusion matrices.

1 Introduction

The main purpose of this study is to implement multiple classifier technique of machine learning and compare their performance on Fashion-MNIST dataset.

Fashion-MNIST dataset contains 70000 images, including 60000 training labeled samples and 10000 testing labeled samples. Each image has resolution of 28×28 and a single channel.

To pre-process the data, Principal Components Analysis (PCA) was used for feature extraction. Beside, normalization is also used for reducing the computation.

Previous work related to Fashion-MNIST dataset were also studied as a reference, which helped with choosing proper model that suitable to this task and has the potential of achieving a good performance.

Three machine learning and deep learning classifiers are introduced and tested in this study, Support Vector Machine(SVM), Random Forest(RF) and Convolutional Neural Network(CNN). A specific comparison among performance of these model are produced.

2 Previous Work

Fashion MNIST is designed to be a replacement for the original MNIST dataset, helping people exercise and understand machine learning algorithms. After this MNIST released, multiple machine learning algorithms have been applied on this dataset and show the different results.

According to the introduction of this dataset, Zalando Research has tested most common machine learning algorithm with multiple parameters[1]. The maximum accuracy of Support Vector Classifier algorithm is 0.897 with C=10 and using RBF kernel. Moreover, it is obvious that there is a gap between RBF kernel and rest kernel, especially the sigmoid kernel performs poorly in this dataset.

Furthermore, the maximum accuracy of Random Forest classifier algorithm is 0.873. In this algorithm, the parameters having the most impact on the performance are number of estimators, criterion and maximum depth. We noticed that some algorithms such as Naïve Bayes are not suitable for image classification, which shows the low accuracy is approximately 0.511. Thus, we give up implementing Naïve Bayes classifier.

On the other hands, some researchers utilize the deep learning model to pursue the better performance. There are many researchers also use this dataset to prototype Neural Network. The accuracy of Convolutional Neural Networks (CNN) two layers along with Batch Normalization and Skip Connection can achieve 0.925[2]. Other CNN algorithm researches also illustrate that CNN with one layer achieve an accuracy of 0.909[3] and the CNN with three layers achieve an accuracy of 0.94[4], which indicates that Convolutional Neural Network is likely to achieve a relatively high accuracy. Compared with CNN, we also found that other deep learning algorithm such as Long-Short Term Memory (LSTM) cannot achieve such high accuracy like CNN, its accuracy is only about 0.89[5].In order to get the better performance, we also implement CNN model.

3 Methods

3.1 Theories

3.1.1 Support Vector Machine

Support vector machine (SVM) is supervised machine learning method which is widely used in classification. SVM is a discriminative classifier formally defined by a separating hyperplane.Theoretically, suppose we use a hyperplane to solve the binary classification issue. The basic of SVM model defines the maximum spacing between the geometric edge region in eigenspace. The SVM a linear classification model, but it can be nonlinear classification via kernel tricks. The learning strategy of SVM is maximize the margin hyperplane and it can be transformed to a convex quadratic programming problem, so the SVM is a convex quadratic programming optimization problem.

Hyperplane In SVM, hyperplane is used to divide the group of points. Hyperplane can be written as:

$$\vec{w} \cdot \vec{x} - b = 0 \tag{1}$$

Primal problem The target of SVM is finding the maximized distance between boundaries, so the optimal hyperplane results is:

$$\min_{w,b} \frac{1}{2}\|w\|^2, \text{ s.t. } y_i\left(w^T x_i + b\right) \geq 1, i = 1, 2, \ldots, m \tag{2}$$

Dual problem Above problem is difficult to solve, so convert it to a dual problem by Lagrange multiplier:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_i x_i^T x_j \tag{3}$$

$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m \tag{4}$$

This function satisfy the Karush-Kuhn-Tucker (KKT) condition which is:

$$\left\{\begin{array}{c} \alpha_i \geq 0 \\ y_i f\left(x_i\right) - 1 \geq |0 \\ \alpha_i\left(y_i f\left(x_i\right) - 1\right) = 0 \end{array}\right. \tag{5}$$

Kernel trick The kernel function is used for converting the nonlinear transformation to the linear problem in special eigenspace in higher dimension.The kernel function is:

$$k\left(\vec{x}_i, \vec{x}_j\right) = \varphi\left(\vec{x}_i\right) \cdot \varphi\left(\vec{x}_j\right) \tag{6}$$

Thus, the function in dual problem has been reconstructed:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_i \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j) \tag{7}$$

$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, 2, \ldots, m \tag{8}$$

Radial Basis Function (RBF) kernel The RBF kernel is the common choice for kernel trick, we use this kernel in our SVM model.

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \tag{9}$$

3.1.2 Random Forest

Random Forest is built by aggregating multiple decision trees together to get the more appropriate and accurate result. Each decision tree in the forest considers a random subset of features when forming a problem and can only access a random set of training data points[9]. The random forest adopts the Bagging sampling strategy.

Samples are selected from the training sample set using Bootstrap sampling, and it selects some of the optimal features as the nodes to split the sample set and build the tree[10]. There are two common division criterion: Gini Impurity and Information Gain. Both criterion are used to judge the uncertainty of the sample dataset. In this study, we only consider the Gini Impurity as the division criterion, because of the lower computation cost.

Gini Impurity If there are classes in the current sample set , the proportion of the sample set of the i-th sample is .Thus, the uncertainty of sample set can be measured by the Gini value:

$$Gini(D) = \sum_{i=1}^{k} p_i(1 - p_i) = 1 - \sum_{i=1}^{k} p_i^2 \tag{10}$$

Thus, the smaller the Gini value is, the uncertainty of sample set is lower. The Gini index is used for finding the optimal division features of the sample set . Assume is a feature in the sample set , so the Gini index can be:

$$Gini - index(D, a) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} Gini(D_i) \tag{11}$$

Then, in order to further get the optimal division feature, we need to find the minimum of the Gini index. The feature of this minimum then be considered as the division standard for next iteration.

3.1.3 Convolutional Neural Network

Convolutional Neural Network in deep learning is a class of neural network which is commonly applied for computer vision tasks. Since the data set Mnist-Fashion are images, convolutional neural network is used as a classifier in this study. Compared to normal neural network, convolutional neural network has advantages in dealing with large input images and preventing overfitting and performance.

Convolutional Neural Network A Convolutional Neural Network is a simple neural network with convolutional layers. When dealing with computer vision tasks, the input images could be really large, which means the input could be a large amount of features corresponding to the number of pixels of the input images. As a result convolutional layers could be used to extract features and reduce the amount of input features feeding into the fully connected layers.

Filter As showed by Figure 1, a convolutional layer consists one or several filters. These filters are used to map the original matrix to a new matrix with some mathematical calculation for feature extraction. In the forward propagation, each filter is convolved across the width and height of the input matrix, with the dot product between the entries of the filter and the input matrix being calculated.[7]

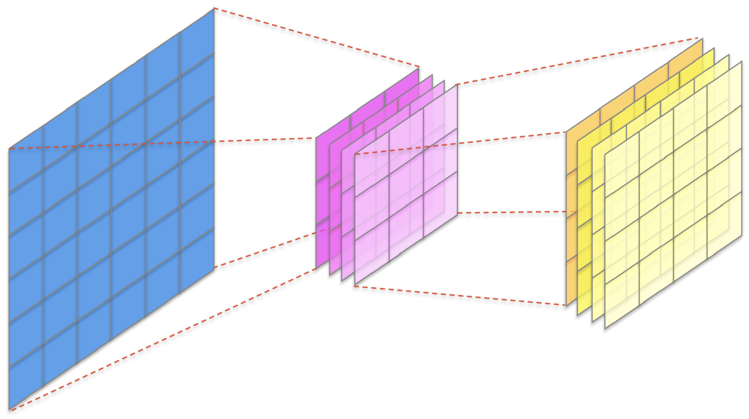


Figure 1: Convolution filter

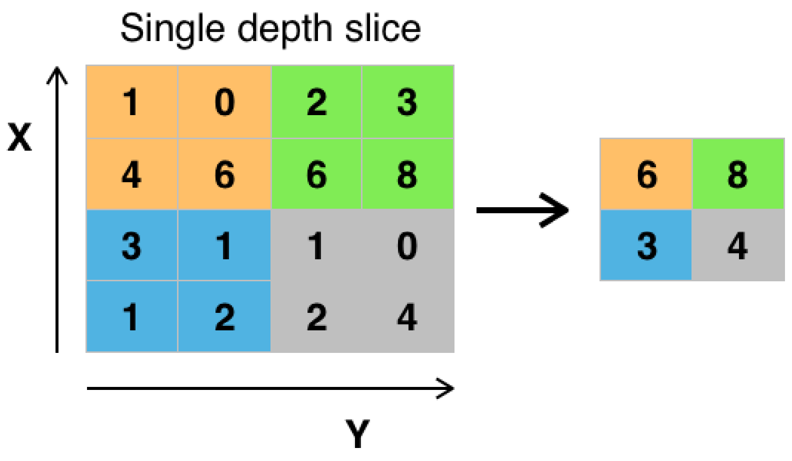


Figure 2: Max Pooling

Pooling After mapping the original matrix to one or more new matrix, pooling is applied before fully connected layer. The basic idea behind is, the exact location of a feature is less important than its location relative to other features. Pooling is used for reducing the input images size and computing cost. As displayed in Figure 2, Max pooling is used in this CNN model since it is the most common pooling method used in Convolutional Neural Networks.

ReLu Activation ReLU is one of the most commonly used activation functions. It gives an output $A = x$ if x is positive and $A = 0$ otherwise.

$$f_a(x) = max(x, 0) \tag{12}$$

ReLU as showed in Figure 3, is also a non-linear activation function which makes it possible to be stacked and deal with non-linear tasks like other activation functions such as Sigmoid and tanh.Compared ReLU with Sigmoid and tanh, the most significant difference is that ReLU has a part of domain that with $f(x) = 0, x < 0$, which means that node is not activated since its output will always be 0.

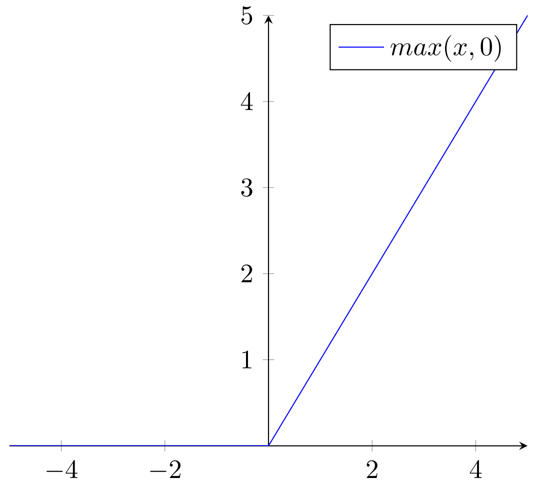


Figure 3: ReLu

3.2 Data processing

3.2.1 Normalization

Image normalization is a key part of ensuring data quality. It can map the data from the original range to the range which is centred by zero. Image normalization is the process of concentrating data by de-averaging.

It reduces the impact if particular features dominate others in the dataset[6]. The formula that is used for standardization is shown below:

$$x = \frac{x - mean}{std} \tag{13}$$

As for convolutional neural network, max normalization is applied. The input is a group of single channel images with pixels ranging from 0 to 255, so we scale the image data from [0, 255] to [0, 1] by dividing 255.0, which can reduce the large computation caused by large numbers.

3.2.2 Principal Components Analysis

Principal Component Analysis(PCA) is the common way to map the data from the high dimension to low dimension. PCA is a feature compression process, but we want the loss of precision to be as low as possible, which means that the most primitive information is preserved during the compression process.

To achieve this goal, we want to spread the expected data points as much as possible. The degree of dispersion can be expressed in terms of variance[8].

The principle of PCA shows as the following: Firstly, calculate the covariance matrix $\sum$ for the matrix $X^{m \times n}$:

$$\sum = \frac{1}{m} X X^T = \frac{1}{m} \sum_{i=1}^{m} X_i {X_i}^T \tag{14}$$

Thus, we can have :

$$\sum = Cov(x_1 1, \cdots, x_m n) = \left\{ \begin{array}{ccc} Cov(x_{11}, x_{11}) & \cdots & Cov(x_{11}, x_{nm}) \\ \vdots & \ddots & \vdots \\ Cov(x_{n1}, x_{11}) & \cdots & Cov(x_{nm}, x_{nm}) \end{array} \right\} \tag{15}$$

Then, using singular value decomposition (SVD) to calculate the eigenvectors for covariance matrix:

$$U, \sigma, V_T = SVD(\sum) \tag{16}$$

Select the first K singular vectors from matrix U and reconstruct the matrix U_rec and the new feature vectors $v_n^{(i)}ew$:

$$U_{reconstruct} = (u^{(1)}, u^{(1)}, u^{(1)}, \cdots, u^{(k)}) \tag{17}$$

$$v_{new}^{(i)} = U_{reconstruct}^T \cdot x^{(i)} \tag{18}$$

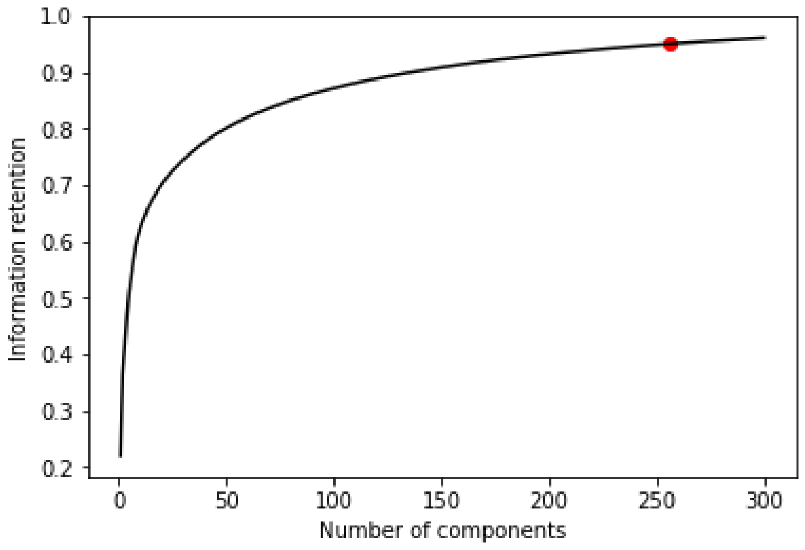


Figure 4: Information retention rate

Figure 4 shows the correlation between information retention rate and the number of components.

In this study, we conduct the PCA on data that is already normalized,and we suppose we only care the information retention rate with 95% (256 components). Thus, eventually, the dimension is reduced from 784 to 256.

4 Experiments and results

4.1 Support Vector Machine

4.1.1 Experiments

For SVM, we have examined two different kernel functions —Radial Basis Function (RBF) and Polynomial Function. We find the result based on RBF kernel is better. Thus, we choose RBF as the kernel function. For SVM based on RBF kernel, we try to tune two parameters to get the best performance:

- C: The parameters exchange the correct classification of the training samples to prevent the marginal maximization of the decision function;
- γ: it determines the distribution of data after mapping to a new feature space. The larger the gamma means the smaller number of support vector. The number of support vectors affects the speed of training and prediction[11].

In order to find the optimized setting for these two parameters, firstly, we examine the SVM model with different C. The table below shows the variance between C and the accuracy. The tuning parameter process based on the first 20000 training dataset.

From Figure 5 and Table 1 we can see when $C = 10$, the model can get the highest accuracy 0.860. But this result is very similar with $C = 1$ and $C = 100$.

Table 1: Accuracy with different C

C	0.001	0.01	0.1	1	10	100
Accuracy	0.103	0.621	0.768	0.855	0.860	0.859

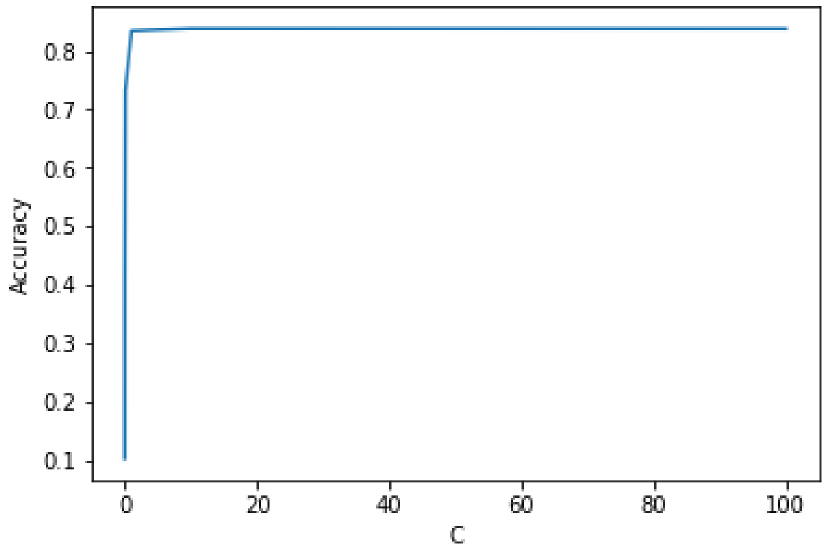


Figure 5: Accuracy and C

Furthermore, as showed in Figure 6, we track the accuracy variance with γ by combining the acceptable C we found. In this case, we try to combine the different $C \in (1, 10, 100)$ with different γ.

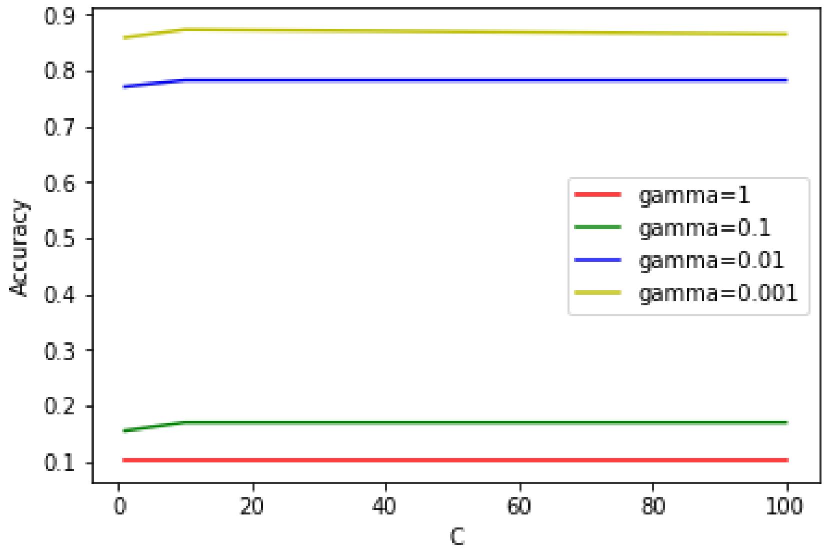


Figure 6: Accuracy with different γ and C

4.1.2 Cross Validation

To avoid to overfitting issue, we use 10-fold Cross Validation to examine the optimal $C = 10$ and $\gamma = 0.001$ we found for the whole 60000 training dataset. Results are displayed in Table 2.

Table 2: SVM 10-fold Cross Validation result

Mean Accuracy	Mean Fitting Time(s)	Score Std	Time Std(s)
0.904	206.9	0.0046	0.7

4.1.3 Result

In order to get the prediction for the test dataset, we train the SVM model with RBF kernel, $C = 10$ and $\gamma = 0.001$. The training time is 220.6 seconds. The time for predicting the label is 75 seconds. The final results of our SVM model is showed in Figure 7 and Table 3.

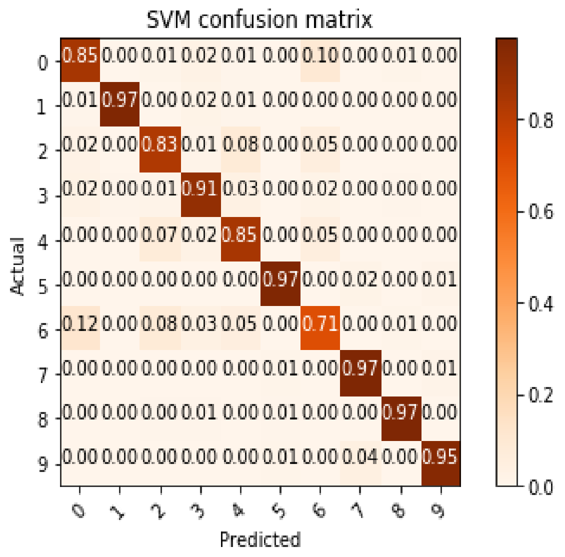


Figure 7: Confusion matrix of SVM result

Table 3: SVM classification report

	precision	recall	f1-score s	upport
0	0.84	0.85	0.84	1000
1	0.99	0.97	0.98	1000
2	0.82	0.83	0.82	1000
3	0.89	0.91	0.90	1000
4	0.83	0.85	0.84	1000
6	0.97	0.97	0.97	1000
6	0.76	0.71	0.73	1000
7	0.94	0.97	0.96	1000
8	0.97	0.97	0.97	1000
9	0.97	0.95	0.96	1000
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

4.2 Random Forest

4.2.1 Experiments

We use Random Forest classification in scikit-learn library for this study. We examine three parameters of RF to get the optimized result.

- `n-estimator`: The number of decision in the forest. Theoretically, the more trees in the forest, the better result we can get. But if there is too many trees in the forest, the time performance will be very low.
- `max_depth`: The maximum depth of each tree.
- `max_features`: Number of features to consider when looking for the best split. Normally, it can be set as `square root` or $\log 2$.

The larger number of `n-estimator` is more time-consuming, so we set the `n-estimator` with the number from 10 to 200, the default value of this parameter in the `scikit-learn\verb` library is 10.The tuning parameter process based on the first 20000 training dataset.

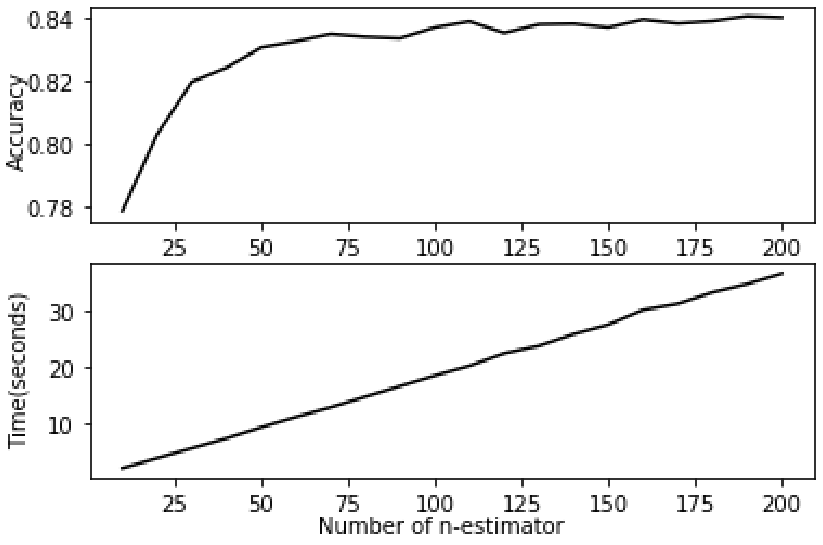


Figure 8: Accuracy and running time of different number of estimators

As we can see from the Figure 8, basically, the more trees we use in the forest, the higher accuracy we can get. However, the fit time also dramatically increase. Also. we find that when the number of estimators is more than 100, the accuracy just grows up slightly, but the fitting time increases very obviously. Especially, when the number of estimators is 200, its fit time is more than twice as 120 estimators, but the accuracy of 200 estimators is only a little higher than 125 estimators, which is only approximately 0.005 difference of accuracy between these two. Thus, we set n-estimator equals to 125.

Furthermore, we examined the impact for the different number of the `max_depth` and the result is showed as Figure 9.

We can see from Figure 9. that when the number of the `max_depth` is more than 20, the accuracy and running time is almost the same value. But when it is less than 20, the accuracy is obviously low. Thus, for this parameter we need to select the value more than 20. Moreover, as we can see from the figure, when `max_depth` equals 50, the standard deviation is the lowest. Thus, we just set it equals to 50.

For the next step, we further tune the last parameter `max_features`. There are two candidates for this parameters that are `square root` or $\log 2$.

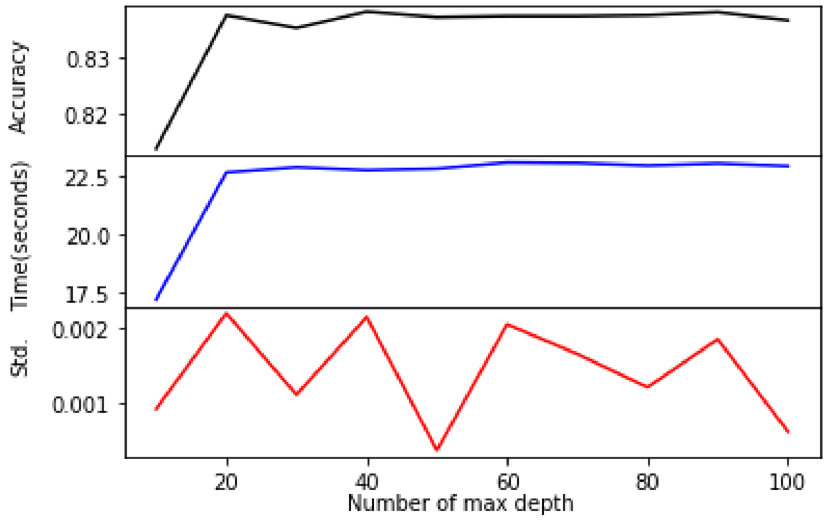


Figure 9: Performance of Random Forest

Table 4: Accuracy and running time with different max features

`max_features`	Mean Accuracy	Mean Fitting Time(s)
$log2$	0.832	12.9
`max_features`	0.838	23.2

The accuracy of `square root` is higher than $log2$. Thus, we select `square root` as the `max_depth`. For random forest classifier the optimal parameter combination is: `max_features` $=$ `sqrt`, `max_depth` $= 50$, $n\_estimator = 125$.

4.2.2 Cross Validation

To avoid to overfitting issue, we use 10-fold Cross Validation to examine the `max_features` $=$ `square root`, `max_depth` $= 50$, $n\_estimator = 125$ for the whole 60000 training dataset. The result is shown below.

Table 5: Random Forest 10-fold Cross Validation result

Mean Accuracy	Mean Fitting Time(s)	Score Std	Time Std(s)
0.867	182.1	0.0034	0.15

4.2.3 Result

To get the prediction for the test dataset, we train the RF model with `max_features` = `sqrt`, `max_depth` = 50, $n\_estimator = 125$. The training time is 210.2 seconds. The time for predicting the label is 0.61 seconds. The final result of our RF model is shown as Figure 10 and Table 6.

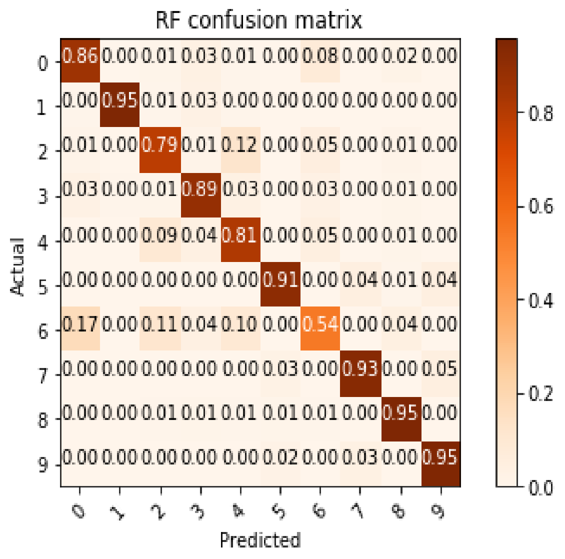


Figure 10: Confusion Matrix of Random Forest

Table 6: Random Forest classification report

	precision	recall	f1-score	support
0	0.79	0.86	0.82	1000
1	1.00	0.95	0.97	1000
2	0.77	0.79	0.78	1000
3	0.85	0.89	0.87	1000
4	0.75	0.81	0.78	1000
6	0.94	0.91	0.92	1000
6	0.72	0.54	0.62	1000
7	0.92	0.93	0.93	1000
8	0.91	0.95	0.93	1000
9	0.91	0.95	0.93	1000
accuracy			0.86	10000
macro avg	0.86	0.86	0.86	10000
weighted avg	0.86	0.86	0.86	10000

4.3 Convolutional Neural Network

4.3.1 Experiments

First we constructed a model which has a similar structure with the previous work mentioned in section 2. Figure shows To reach a possible best performance, experiments regarding to CNN parameters including learning rate, optimizer, dropout rate and normalization. We used Control Varietes for discovering the best combination of parameters.

From Table 7 we can see, 8 different models were tested. Model 4 was tested to be the one with best performance, including accuracy and running time.

When training number of epochs is 30, overfitting problem occurs but not significant. For preventing potential overfitting when the number of epochs grows, we choose a learning rate 0.6.

Table 7: CNN validation comparison

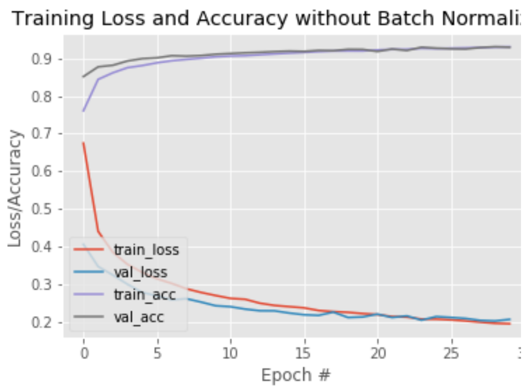
Model	Opt.	LR	Dropout	BN	Acc.	Time(s)
1	Adadelta	1.0	0.0	False	0.92	112.25
2	Adadelta	1.0	0.25	False	0.93	124.2
3	Adadelta	1.0	0.4	False	0.93	125.91
4	Adadelta	1.0	0.6	False	0.93	125.29
5	Adadelta	1.0	0.6	True	0.93	163.34
6	SGD	0.01	0.6	False	0.90	116.13
7	SGD	0.05	0.6	False	0.91	115.09
8	SGD	0.1	0.6	False	0.92	114.35

Also from Table 7, comparison among model 4, 5, 6, 7, 8 suggests that optimizer Adadelta has a better accuracy but also with a sightly increase of running time, however since it is still in an acceptable range of time, we chose to use Adadelta as model optimizer, and the result also proves that Adadelta has advantages compared to SGD.

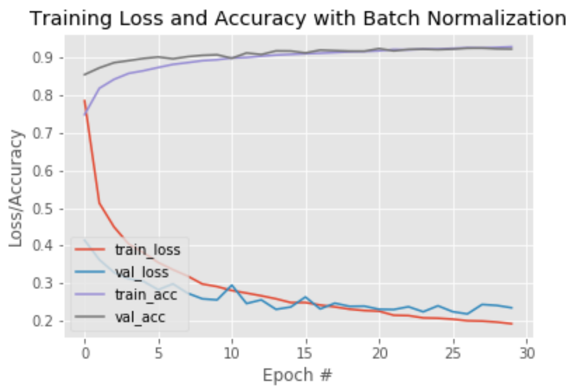
As for learning rate, since default $learningrate = 1$ is recommended by Keras official API, we focused on the test on learning rate when using SGD as optimizer. There is no strong evidence that learning rate has a remarkable influence on this specific data and task regarding to the comparison between model 6, 7, 8 showing in Table 7.

Besides, the effect of batch normalization was also tested by the comparison of model 4 and 5. From the accuracy comparison, the effect of batch normalization is not noticeable. So we further perform a comparison of the loss decay gradient between model 4 and 5. By comparing Figure 11, it is found that with Batch Normalization, the train loss curve is smoother than the curve without, which is a proof to the theory of Batch Normalization.

However, regarding to this specific task and its overall performance, we decided to not use Batch Normalization since it does not have significant improvement to accuracy but the running time is 30% longer than the model not using Batch Normalization.



(a) without Batch Normalization



(b) with Batch Normalization

Figure 11: Training Process

4.3.2 Cross Validation

To validate the performance of the model and prevent potential overfitting, 10-fold cross validation was performed and the result is showing in Table 8.

Table 8: CNN 10-fold Cross Validation result

Model	Accuracy	Std
CNN	92.98%	$+/-0.42\%$

4.3.3 Result

The final model reaches a testing accuracy of 0.93 and running time of 125.29 seconds with parameters specifying by Table 9.

Table 9: CNN final model

Model	Opt.	LR	Dropout	BN	Acc.	Time(s)
4	Adadelta	1.0	0.6	False	0.93	125.29

Model structure is as showed in Figure 12.

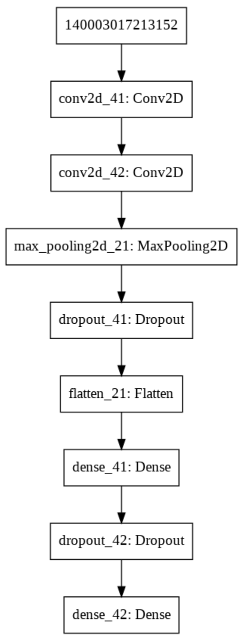


Figure 12: Final model structure

The training process is showed by Figure 11(a), and prediction result is displayed by Table 10.

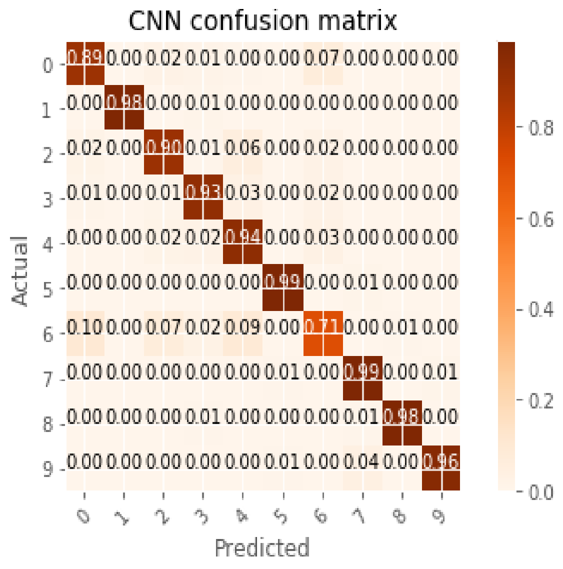


Figure 13: CNN Confusion Matrix

Table 10: CNN classification report

	precision	recall	f1-score	support
0	0.87	0.89	0.88	1000
1	0.99	0.98	0.99	1000
2	0.83	0.93	0.87	1000
3	0.92	0.93	0.92	1000
4	0.91	0.83	0.87	1000
6	0.99	0.99	0.99	1000
6	0.80	0.76	0.78	1000
7	0.96	0.98	0.97	1000
8	0.99	0.98	0.99	1000
9	0.99	0.96	0.97	1000
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

5 Comparison

In this section, we compare the accuracy and time performance among the three implemented models. The data which is visualized in the pictures are based on our previous experiment. Comparison among accuracy fitting time and predicting time of each model is displayed on figure 14.

Hardware Environment All the experiments are performed under the GPU version of Google CoLab Running time showing by Table 11.

Table 11: Hardware environment

Platform	CPU	GPU	Memory
Google CoLab	Intel(R) Xeon(R) CPU @ 2.30GHz	Tesla K80	12GB

Accuracy The figure 14 suggest that CNN has the highest accuracy of 93%, followed by SVM with accuracy of 90%, and Random Forest has the lowest accuracy of 86%.

Fiiting time As for fitting time, CNN has the shortest training time of 125.29 seconds which is noticeably lower shorter than both Random Forest and SVM.

Predicting time SVM has the longest predicting time of 75 seconds which is significantly longer than CNN and Random Forest, 1.44 and 0.6 seconds separately.

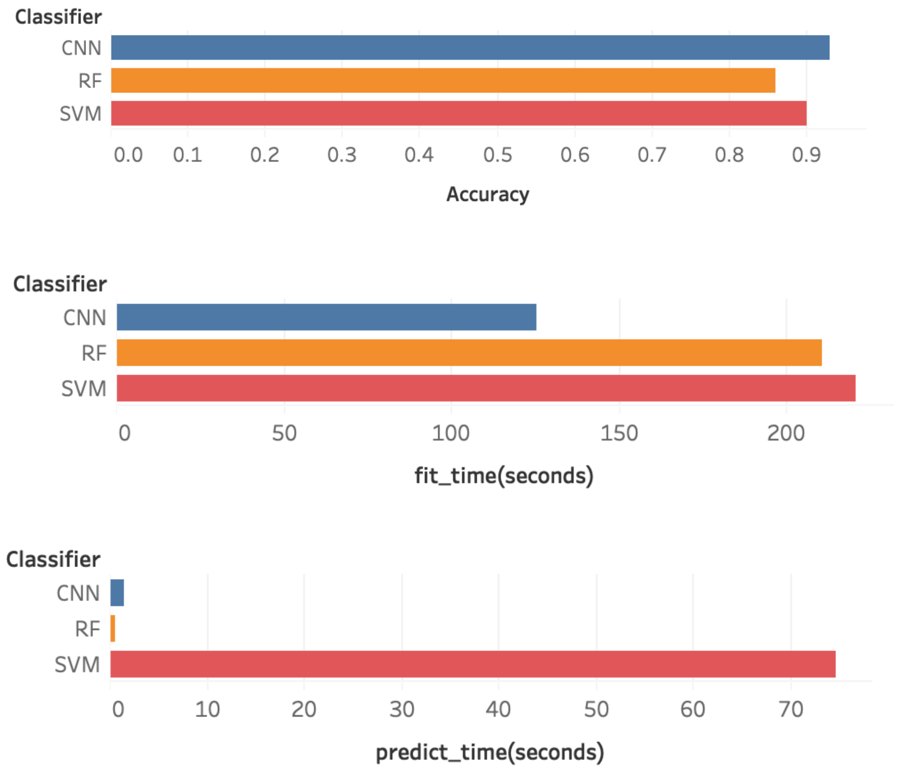


Figure 14: Performance Comparison

6 Conclusion

In conclusion, in this study, we have examined three different machine learning classifiers which are Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN).

For the pre-processing stage, we use the two image pre-processing techniques-normalization for concentrating data by de-averaging and PCA for reducing the dimension, thereby reducing the computation cost. As for the classifier performance, in our experiment, CNN has the best performance with both the highest accuracy of 93% and lowest time cost of 125.29s for training the network. SVM also has the acceptable performance with 90% accuracy. However, random forest is the only one that the accuracy is less than 90% with only 86%.

Discussion In the future, we need to use more advanced pre-processing techniques and better adjust the parameters, if we are keen to achieve higher accuracy. Moreover, in order to further reduce the time cost, especially when the dataset and computation workload is very massive, we will consider to use Spark Framework which can run machine learning algorithm as parallel in large clusters.

References

[1] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.

[2] S. Bhatnagar, D. Ghosal, M.H. Kolekar, "Classification of fashion article images using convolutional neural networks", ICIIP, pp. 1-6, 2017.

[3] J. Brownlee, "How to Develop a Deep Convolutional Neural Network From Scratch for Fashion MNIST Clothing Classification", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/.

[4] A. Rosebrock, "Fashion MNIST with Keras and Deep Learning - PyImageSearch", PyImageSearch, 2019. [Online]. Available: https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/.

[5] Zhang, K. LSTM: An Image Classification Model Based on Fashion-MNIST Dataset, 2017

[6] S. Srinidhi, "Why do we need feature scaling in Machine Learning and how to do it using SciKit Learn?", Medium, 2018. [Online]. Available: https://medium.com/@contactsunny/why-do-we-need-feature-scaling-in-machine-learning-and-how-to-do-it-using-scikit-learn-d8314206fe73.

[7] "Convolutional neural network", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network. [Accessed: 30- May- 2019].

[8] Z. Jaadi, "A step by step explanation of Principal Component Analysis", Towards Data Science, 2019. [Online]. Available: https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2. [9] "SVC Parameters When Using RBF Kernel", Chrisalbon.com, 2017. [Online]. Available: https://chrisalbon.com/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/.
[10] W. Koehrsen, "Random Forest Simple Explanation", Medium, 2019. [Online]. Available: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d.
[11] N. Donges, "The Random Forest Algorithm", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

Appendix

Latex Format

This document is formatted following *Neurips 2019* standard.

Run Instruction

There are two types of code files, Type one: "****Training.ipynb". The suffix of the filename is "Training.ipynb". This kind of files contain the details about process of tuning the parameters and 10-fold cross validation. Please DO NOT run this type of codes. It will spend a lot of time.

Type two "****Trained.ipynb". The suffix of the filename is "Trained.ipynb". This type of files refers to model that we already have fully tune the parameter, it can be directly run to predict the output. Please RUN this type of files.