Contents

1 ABSTRACT **1**

2 INTRODUCTION **1**

2.1 Problems intended to be solved and importance 1

2.2 Overview of data sets and methods 1

2.3 Previous work 2

3 METHODS **3**

3.1 SVM(Support Vector Machine) 3

3.2 Random forest 3

3.3 CNN(Convolutional neural network) 5

3.3.1 Structure of convolutional neural network) 5

3.3.2 Batch Normalization 5

3.3.3 Dropout 5

3.4 Data pre-processing 6

3.4.1 Normalization 6

3.4.2 Principal Component Analysis (PCA) 7

4 Experiments and Discussion **8**

4.1 SVM(Support Vector Machine) 8

4.2 Random Forest 10

4.3 CNN(Convolutional neural network) 12

4.3.1 CNN Model 1 (CNN with Batch Normalization but No Dropout) 12

4.3.2 CNN Model 2 (CNN without Batch Normalization and No Dropout) . . 13

4.3.3 CNN Model 3 (CNN with Batch Normalization and Dropout) 15

4.4 Evaluation indicators 17

5 Conclusion and Future Work **17**

5.1 Conclusion 17

5.2 Reflection 18

5.3 Future Work 18

6 APPENDIX **19**

6.1 Hardware and Software Specifications 19

6.2 Instructions for running the code 19

List of Figures

1 Partial view of the dataset 2
2 Schematic diagram of SVM principle 3
3 Diagram of the implementation process of Random Forest 4
4 Convolutional neural network structure diagram 5
5 Schematic diagram without normalization 6
6 Schematic diagram with normalization 6
7 Diagram of the process of finding PC1 7
8 Diagram of PCA variance ratio 7
9 Schematic diagram of accuracy and runtime of SVM 8
10 Schematic diagram of accuracy and runtime of SVM 9
11 SVM classified evaluation chart 9
12 SVM confusion matrix diagram 10
13 Schematic diagram of accuracy of Random Forest 11
14 Random Forest classified evaluation chart 11
15 Random Forest confusion matrix diagram 11
16 The Accuracy, Precision and Recall rate of Model 1 in train data 12
17 The Confusion matrix Model 1 in train data 12
18 The loss value changes of Model 1 in train data 13
19 The output data of Model 1 in train data 13
20 The Accuracy, Precision and Recall rate of Model 2 in train data 14
21 The Confusion matrix Model 2 in train data 14
22 The loss value changes of Model 2 in train data 14
23 The output data of Model 2 in train data 14
24 The Accuracy, Precision and Recall rate of Model 3 in train data 15
25 The Confusion matrix Model 3 in train data 15
26 The loss value changes of Model 3 in train data 16
27 The output data of Model 3 in train data 16
28 The output of CNN model 1 in test dataset 16
29 The confusion matrix of CNN model 1 in test dataset 17
30 Schematic diagram of importing data 19

List of Tables

1 ABSTRACT

With the rapid development of machine learning and deep learning techniques, more and more techniques are being developed to solve real-life image classification problems, and with the many different techniques available, it is sometimes difficult to decide which one to choose for practical purposes. For this reason, this report uses the EMNIST ByClass Dataset as the basis for classifying this dataset by applying machine learning and data mining with Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN). (SVM), Random Forest (RF) and Convolutional Neural Network (CNN) to classify this dataset, and analyse, discuss and compare the performance of each classification algorithm. The results show that Convolutional Neural Network (CNN) performs best in terms of accuracy on the test set we used.

2 INTRODUCTION

2.1 Problems intended to be solved and importance

The focus of this research is on the recognition and classification of handwritten characters through machine learning algorithms to predict the correct characters, as image and text recognition technology is increasingly used in our daily lives today. By tapping these handwritten texts we can easily make calls, look up directions, send emails or translate texts, greatly enhancing the convenience of our lives. In the field of driverless vehicles, there is also an urgent need to improve the accuracy of the current intelligent recognition of road signs and pedestrian vehicles in order to make driverless technology safer and more convenient for us in the future.

Therefore, in this study, through the knowledge of machine learning and data mining that we have learnt in class, we will perform suitable pre-processing on the given data set, find an effective classification algorithm to carry out the classification, and adjust the parameters through training to obtain a higher accuracy rate. A good foundation is laid.

2.2 Overview of data sets and methods

For this report, we chose EMNIST ByClass as the dataset used in our study, which is a 62-category, 814,255-character dataset in 28□28 pixel format, which was converted from handwritten character numbers from NIST Special Database 19.[2]The following figure shows the partial effect of the dataset.

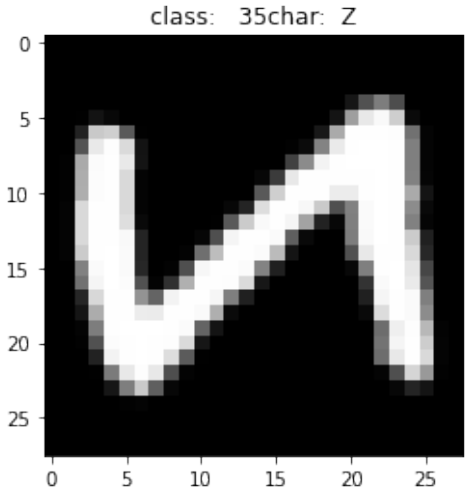

Figure 1: Partial view of the dataset

Three algorithms, Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN), were used to evaluate the performance of each classification algorithm in terms of accuracy, recall and confusion matrix, fine-tuning the parameters to achieve better performance for each algorithm on a given dataset.

2.3 Previous work

Aboozar Taherkhani and Georgina Cosma have combined AdaBoost and convolutional neural networks (CNN) in their research paper ”AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning”, they combine AdaBoost and convolutional neural networks (CNNs) to design a new machine learning method called AdaBoost-CNN. This new machine learning method can process large imbalanced datasets with high accuracy, reducing the computational cost of AdaBoost when processing large datasets by reducing the number of learning cycles required by the component estimator, improving progress and reducing training time compared to traditional training methods.[8]

Through their research, we have learnt some lessons about specific classification processes for different datasets and about combining different machine learning algorithms for research. Here, we summarise the methods used in their study that we had not thought of and provide a good basis for our subsequent research on related classifications.

1. Training the dataset using the RMsprop optimizer with an adaptive learning rate.

2. Controlling CNN learning with a large number of training parameters by sample weights to prevent overfitting of the training data.

3. When dealing with unbalanced data, a weighted loss function can be used in conjunction with a CNN to build a CNN with a weighted loss function.

4. When using deep learning methods for large datasets, the number of training periods for all estimators should be reduced in order to reduce the high computational cost.

3 METHODS

3.1 SVM(Support Vector Machine)

Support Vector Machine is a supervised learning method, generally used to deal with classification problems. In a general sense, SVM is a binary linear classifier, which is based on the principle of finding a point nearest to the classification hyperplane on the positive and negative sides of the classification hyperplane, to maximize the sum of the distances between these two points from the classification hyperplane. The diagram below shows the principle of svm.

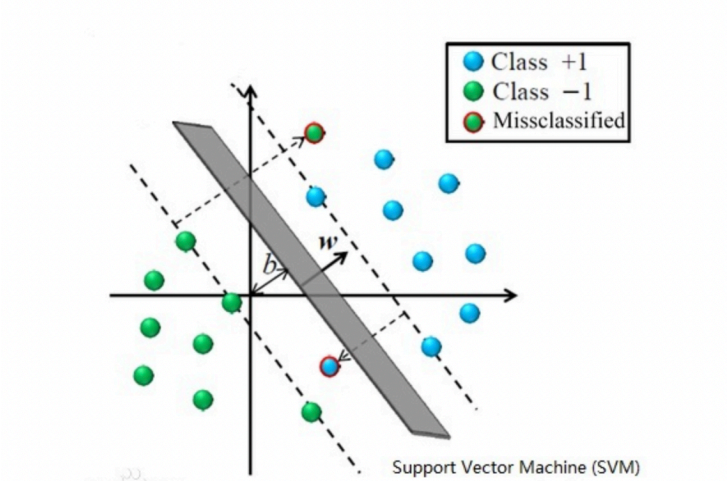

Figure 2: Schematic diagram of SVM principle

The SVM classification strategy focuses on choosing the hyperplane with the largest margins in order to prioritise the correct classification of the training data and then to set as much space as possible for the noise to improve the robustness of the classifier and to prevent overfitting, and the SVM itself uses kernel functions to transform linearly indistinguishable data into a higher dimensional feature space to make it linearly distinguishable.[6]

The reason we chose the SVM algorithm for classification processing is that the algorithm can be designed to classify different types of data by combining different kernel functions, as well as image data or other types of data with different structures. The algorithm is often used in image classification and recognition, and we do this in sklearn by calling the SVC model and setting 'rbf', The results show that the linear kernel performs best.

3.2 Random forest

Random Forest is an integrated algorithm consisting of decision trees, it is a supervised learning algorithm based on if-then-else rules, and it fits our intuitive thinking and is easy for us to

interpret. [7]It consists of a number of decision trees, but the different decision trees are not related to each other. When we perform this classification task, we input a new sample and it lets each decision tree in the forest judge and classify it separately, so that each decision tree gets its own classification result, and finally the random forest summarises which of the decision trees has the most classifications, and it takes this result as the final result. The diagram below shows the process of implementing the random forest.

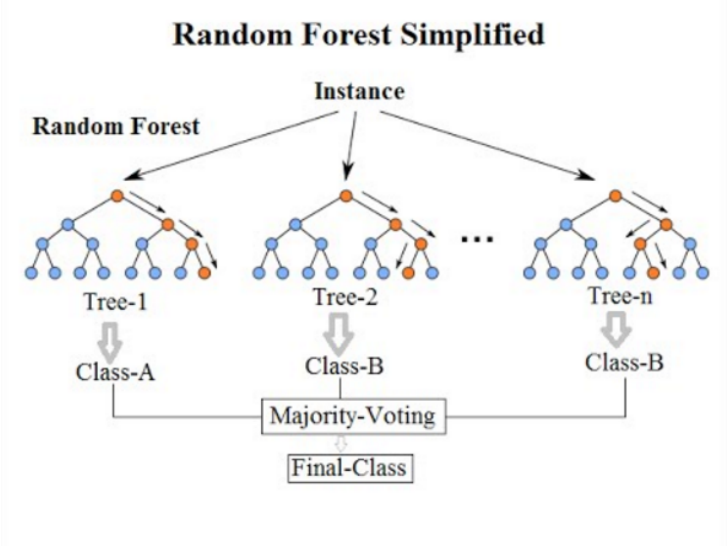

Figure 3: Diagram of the implementation process of Random Forest

The Random Forest algorithm operates with high dimensionality and many features, without dimensionality reduction, allowing us to determine the importance of the features, and it is faster to train and less prone to overfitting, while balancing these errors for the unbalanced EMNIST ByClass dataset we are using. Even if a large proportion of the features are lost, the random forest can still maintain a certain level of accuracy.

The reason we designed to use the random forest algorithm is that it is a more intuitive process to implement and fits our thinking perception. We first design a randomly selected sample to train the decision tree, then the system will randomly select the attributes in each sample to do the node splitting attribute, after which this step is repeated a lot until no more splitting can be done, and eventually a large number of decision trees will be built and a random forest will be formed. This process of our choice in python sklearn library to implement, the experiment at first we through RandomizedSearchCV to automatically find the appropriate parameters, but this will make the computer to carry out a lot of repeated verification, time-consuming, and finally we selected the appropriate parameters after the training, effectively reducing the consumption of computer resources.

3.3 CNN(Convolutional neural network)

3.3.1 Structure of convolutional neural network)

Convolutional neural network was invented in 1998 and is essentially a multilayer perceptron. The advantage of it is more obvious when the input to the network is an image, which allows the image to be directly used as the input to the network. When the input of the model is 2D images, it avoids the complex process of feature extraction and data reconstruction in traditional recognition algorithms.[4] For example, the network can extract the features of the image by itself including color, texture, shape and topology of the image. Besides, it also has good robustness and operational efficiency in the problem of processing two-dimensional images. Most of convolutional neural networks have same structure including Convolutional layer, Pooling layer, ReLU layer (Rectified Linear Units layer) and Fully-Connected layer.

The following image shows the structure of Convolutional neural network used in this study.

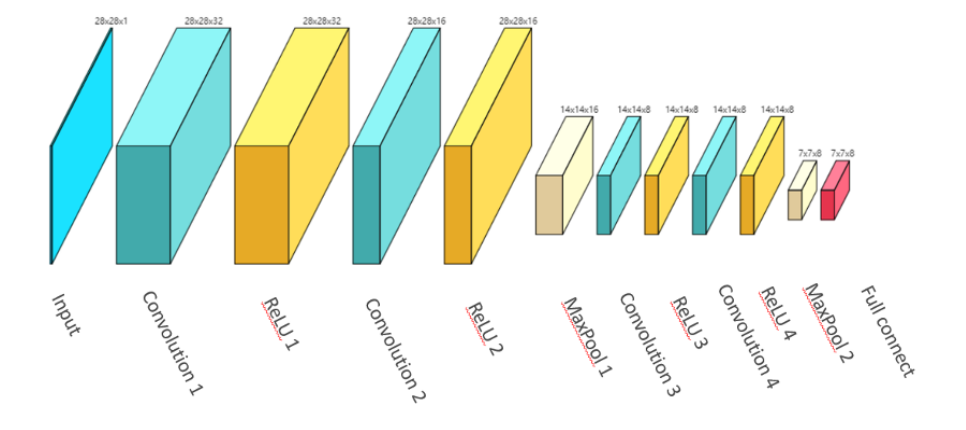

Figure 4: Convolutional neural network structure diagram

3.3.2 Batch Normalization

Batch Normalization is similar to common data normalization, which is a way to unify scattered data and to optimize neural networks. It can let the input of each layer have a same average value and variance, which ignores the influence of large data to the whole model.

3.3.3 Dropout

In machine learning models, if the model has too many parameters and few training samples, the trained model will easily produce the phenomenon of overfitting. The overfitting problem is often encountered when training neural networks. In 2012, Hinton proposed Dropout in his paper. (Hinton et al,2012) In forward propagation, it can let the activation value of a certain neuron stop working with a certain probability p.[9] This makes the model more generalizable because it does not depend too much on certain local features.

3.4 Data pre-processing

In machine learning and data mining projects, the time spent through data processing can be significant, and the quality of the data also determines the strength of the prediction and generalisation ability of the machine learning algorithms, as real data often contains a large number of missing values and a lot of noise, so we should pre-process the data appropriately as a way to get more standard and clean data, the following is a description of the pre-processing techniques used in this experiment.

3.4.1 Normalization

Since we are using a Convolutional Neural Network (CNN) algorithm, it is important to pre-process the data using normalisation. The purpose of normalisation is to limit the pre-processed data to a certain range, e.g. between [0,1], so as to eliminate the negative effects of odd sample data on the experimental results.

If the data is not normalised, the objective function will become ”flat” due to the large difference in values between the different features in the feature vector, which will make the direction of the gradient deviate from the direction of the minimum value during gradient descent,[3] resulting in many detours and long training times, as shown in the figure below.

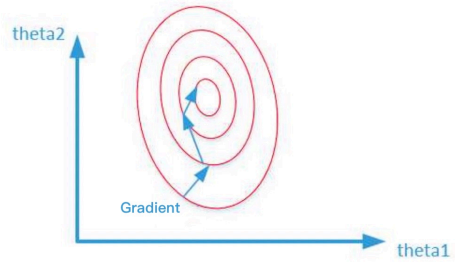

Figure 5: Schematic diagram without normalization

If the data is normalised, the objective function will be more ’round’, which will also make training significantly faster and less circuitous, as shown in the figure below.

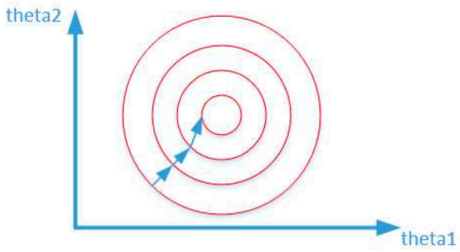

Figure 6: Schematic diagram with normalization

3.4.2 Principal Component Analysis (PCA)

In machine learning, Principal Component Analysis (PCA) is a very widely used algorithm for dimensionality reduction of data. The underlying idea is to map the original n-dimensional data to a k-dimension that is less than n-dimensional, where the k-dimension is the new orthogonal characteristic, also known as the principal component. [5]In our practical experiments, the aim of doing PCA is to find a series of principal components that best represent the variability in the data set, for example, the diagram below shows the process of finding PC1.

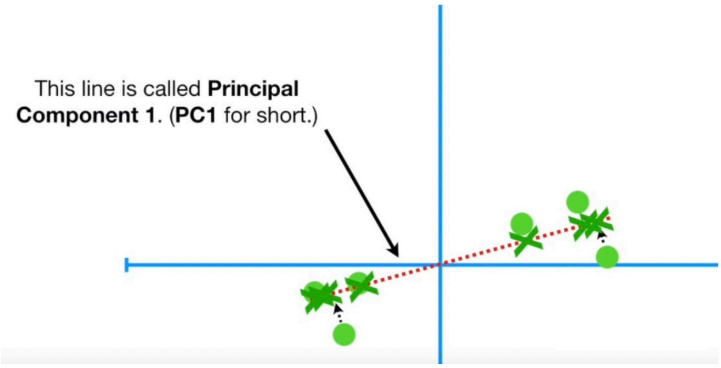

Figure 7: Diagram of the process of finding PC1

As can be seen from the diagram, we find a line in space that maximises the sum of the squared distances from the projection point to the origin when all the points in space are projected onto this line, then the dataset contains the maximum amount of information on this line. We repeat this process for the dataset, so that the original high-dimensional data becomes a mapping of points on each principle component, completing the process of dimensionality reduction of the dataset.

In this experiment, we choose the number of components to use after the pca dimensionality reduction process, and we should strike a balance between dimensionality and principal component variance.

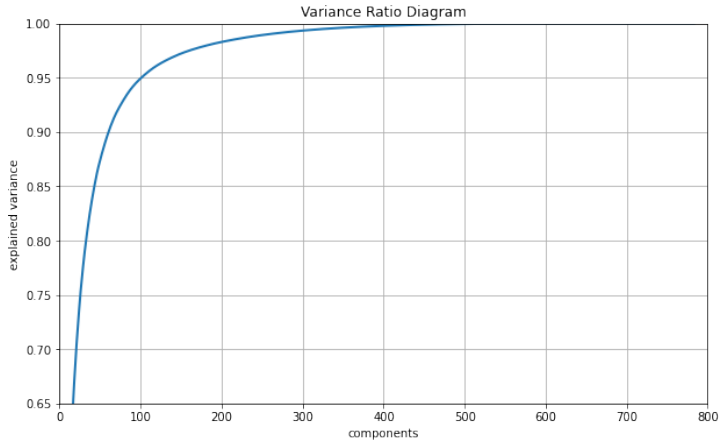

Figure 8: Diagram of PCA variance ratio

The x-axis represents the dimensionality of the data and the y-axis represents the amount of variation explained. From the figure, we can see that when we choose 100 principal components, we can explain more than 95% of the variation.

4 Experiments and Discussion

In this report, there are 814,255 data in the original EMNIST ByClass dataset. Due to the limited performance of our computer, it takes a long time to process such a large amount of data and may cause the computer to crash, so we chose 50,000 of them for our experiments. We divided the 50,000 data into 40,000 training sets and 10,000 test sets, and used three classification algorithms, namely Support Vector Machine (SVM), Random Forest (RF) and Convolutional Neural Network (CNN), to conduct the experiments respectively, and the specific training process and evaluation process are described below.

4.1 SVM(Support Vector Machine)

Support vector machine SVM is a binary classification model which uses a hyperplane to segment the data and is more suitable for processing high-dimensional data. When we perform the SVM algorithm for classification, we need to consider three parameters, C, kernel and gamma, where the hyperparameter C is the penalty coefficient of the error term, the larger the value of C, the higher the accuracy in the training set, the generalization ability will be reduced, and The kernel is the type of kernel function used in the algorithm. In our experiments, the kernel function used is 'rbf': Gaussian kernel function and 'linear': linear kernel function, which in turn yielded different check-all rates. auto, representing its value as the reciprocal of the number of sample features. We first experimented the parameter C on [1,3,5] and obtained the following results.

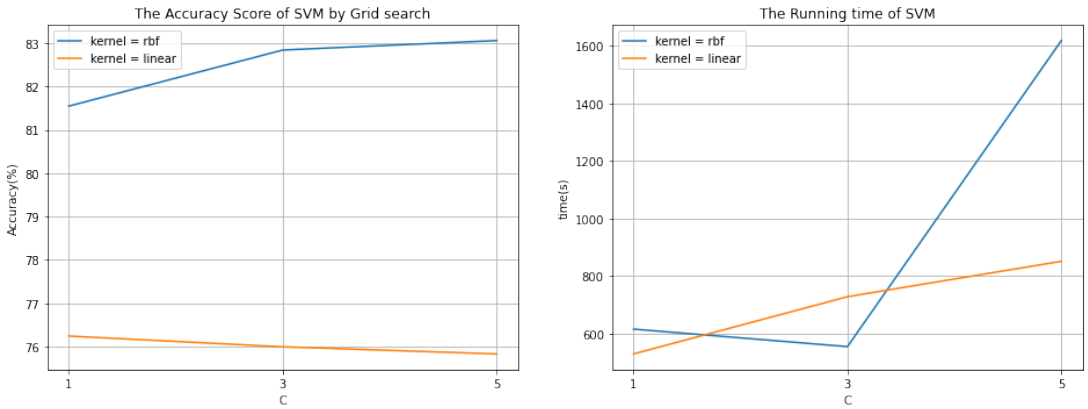

Figure 9: Schematic diagram of accuracy and runtime of SVM

As can be seen from the graph above, there is little difference in accuracy when C is 3 and 5, but when C is 5, the training time of the model becomes significantly longer, so we conducted

further experiments on [1,2,3] using the reduced approach in order to find the optimal parameter C. We then used ten-fold cross-validation in order to find the better parameters and test the accuracy of the algorithm. Ten-fold cross-validation was chosen because it has been shown that a large number of experiments using different learning techniques with a large dataset have shown that ten-fold cross-validation is an appropriate choice for obtaining the best error estimates, but this is not absolute and is only generally accepted. The experimental results are shown in the figure below.

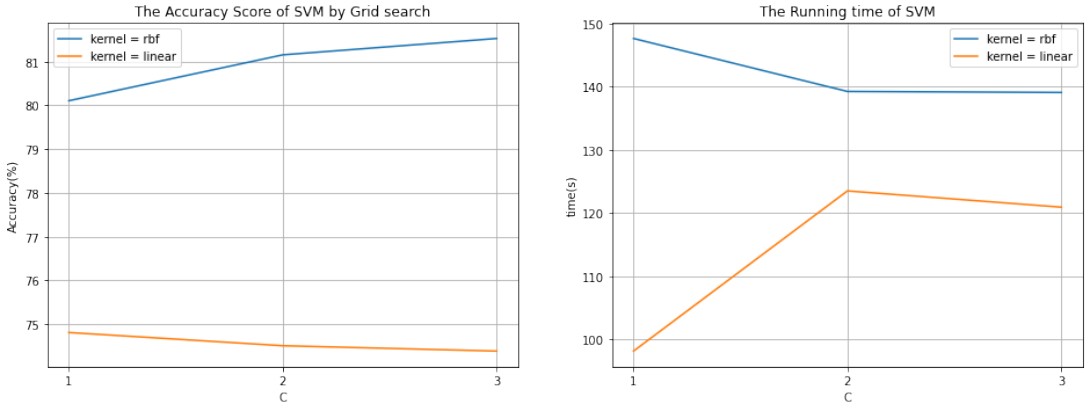

Figure 10: Schematic diagram of accuracy and runtime of SVM

As can be seen from the graph, the highest accuracy of over 83% is achieved when C=3 and kernel=rbf, but at the same time, it has a relatively long running time. So the final experimental results show that the best estimator for the SVM classifier is 'C': 3, 'gamma': 'auto', 'kernel': 'rbf'. At the end of the training, we evaluated the SVM classification results using the accuracy, precision, recall and confusion matrix and the results are shown in the figure below.

```
Accuracy:  29.26%, Recall: 0.293, Precision: 0.293
[[177   0   2 ...   0   1   0]
 [  0  84   1 ...   0   0   0]
 [  6   9 148 ...   1   2   5]
 ...
 [  0   6   1 ...   5   4   1]
 [  0   0   0 ...   0   0   0]
 [  0   0   9 ...   2   0  15]]
```

Figure 11: SVM classified evaluation chart

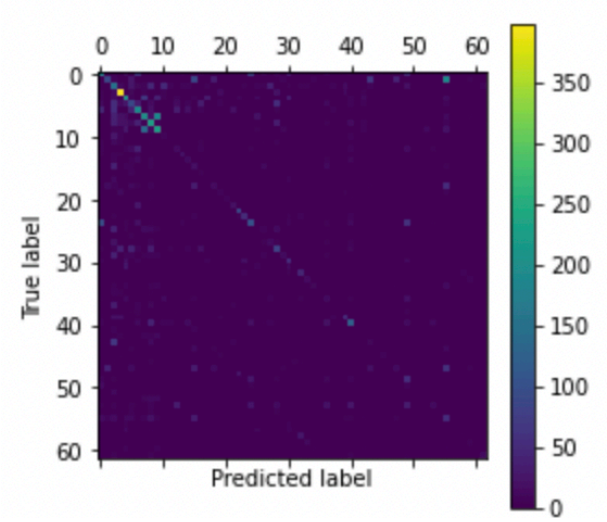

Figure 12: SVM confusion matrix diagram

4.2 Random Forest

The random forest algorithm is an algorithm that integrates multiple trees through the idea of integrated learning. [1]We choose three parameters to adjust (max_depth, max_features, n_estimators). 'max_depth' indicates the maximum depth of the tree, and generally speaking when the data 'max_depth' indicates the maximum depth of the tree, generally when there is less data or fewer features, we can ignore this value, but in this experiment the dataset is large, so we limit the training to between [10, 20,30]; 'max_features' is the maximum number of features considered in the partitioning of the random forest, and various types of values can be used. 'auto' and 'sqrt' are to indicate taking the open square of the total number of features; 'n_estimators' is to indicate the maximum number of iterations of the weak learner when When the value of 'n_estimators' is too small, the model tends to be underfitted, and when its value is too large, the computational effort increases exponentially, and after its value reaches a certain number, the model improvement obtained by increasing its value will be small. Therefore, it is a good strategy to narrow the range based on the results of previous experiments.

Next, we used a grid search and performed a ten-fold cross-validation to find a better combination of parameters, and the resulting best estimator for the random forest classifier was 'max_depth': 30, 'max_features': 'sqrt', 'n_estimators': 150. This is shown in the figure below.

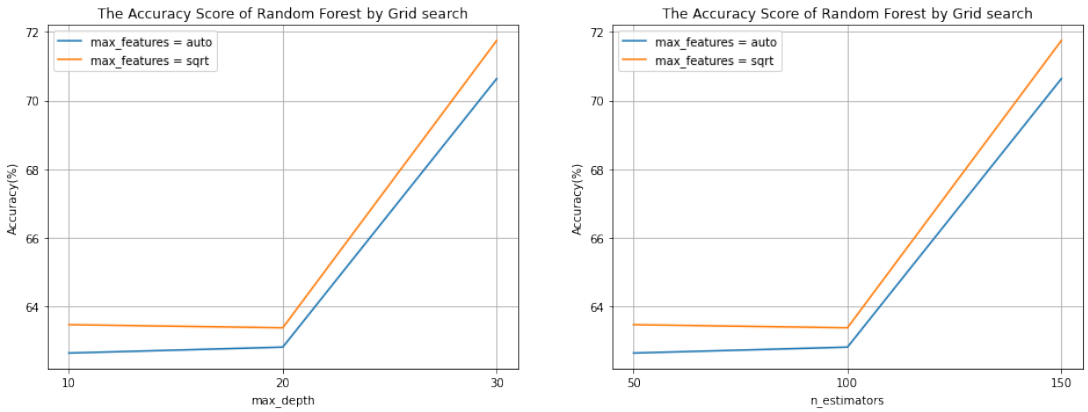

Figure 13: Schematic diagram of accuracy of Random Forest

As can be seen from the graph, the training accuracy of the random forest increases as the values of 'max_depth' and 'n_estimators' increase. Among the values we set, when 'max_features' is 'sqrt", and max_depth' and 'n_estimators' are 30 and 150 respectively, the model obtains the the highest accuracy rate, nearly 72%.

At the end of the training, we evaluated the random forest classification results using the accuracy, precision, recall and confusion matrix, and the results are shown in the figure below.

```
accuracy = accuracy_score(y_pred, label_test)
precision = precision_score(y_pred, label_test, average='micro')
recall = recall_score(y_pred, label_test, average='micro')

print("Accuracy:  %.2f%%, Recall: %.3f, Precision: %.3f"  % (accuracy*100, recall, precision))
```

```
Accuracy:  36.85%, Recall: 0.368, Precision: 0.368
```

Figure 14: Random Forest classified evaluation chart

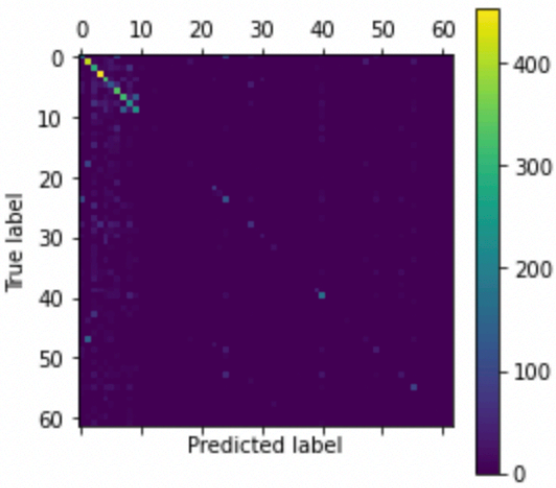

Figure 15: Random Forest confusion matrix diagram

4.3 CNN(Convolutional neural network)

There are 3 kinds of convolutional neural network models in this study, the differences of them are in the fellow table.

Table 1 Difference diagram of 3 CNN models

Number	Batch Normalization	Dropout
Model 1	Yes	No
Model 2	No	No
Model 3	Yes	Yes

All of them have the same network structure, learning rate, weight decay, batch size, optimizer and loss function. Besides, they has same train data and test data.

4.3.1 CNN Model 1 (CNN with Batch Normalization but No Dropout)

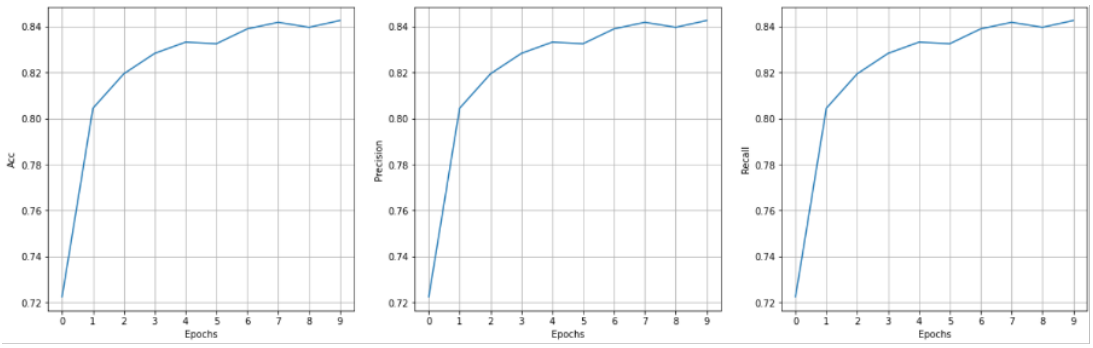

Figure 16: The Accuracy, Precision and Recall rate of Model 1 in train data

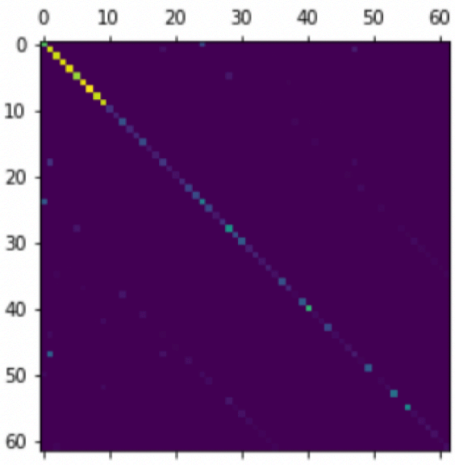

Figure 17: The Confusion matrix Model 1 in train data

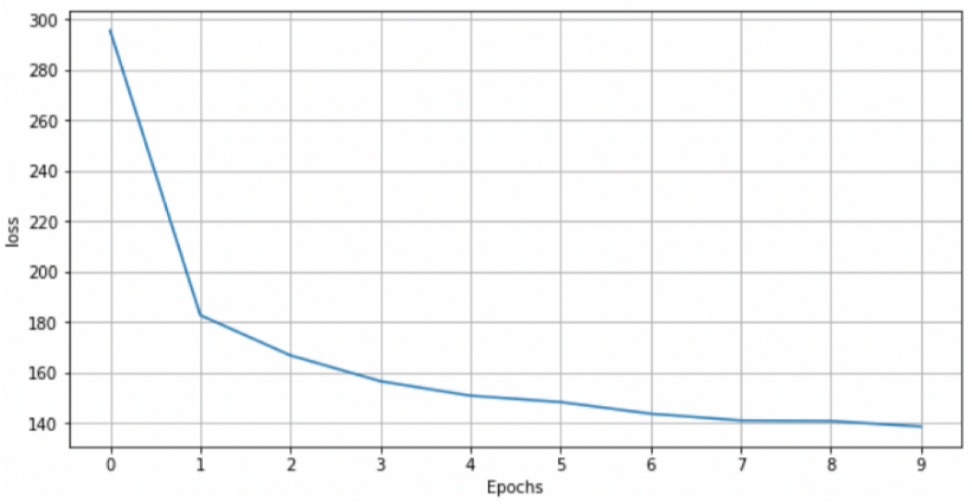

Figure 18: The loss value changes of Model 1 in train data

```
Epoch: 1, loss: 295.58019, Acc: 72.25, precision:  72.252, recall: 72.252
Epoch: 2, loss: 182.81378, Acc: 80.44, precision:  80.445, recall: 80.445
Epoch: 3, loss: 166.85905, Acc: 81.94, precision:  81.942, recall: 81.942
Epoch: 4, loss: 156.60865, Acc: 82.83, precision:  82.830, recall: 82.830
Epoch: 5, loss: 150.86077, Acc: 83.32, precision:  83.317, recall: 83.317
Epoch: 6, loss: 148.34328, Acc: 83.25, precision:  83.250, recall: 83.250
Epoch: 7, loss: 143.73224, Acc: 83.89, precision:  83.895, recall: 83.895
Epoch: 8, loss: 141.03838, Acc: 84.18, precision:  84.180, recall: 84.180
Epoch: 9, loss: 140.83289, Acc: 83.96, precision:  83.962, recall: 83.962
Epoch: 10, loss: 138.63761, Acc: 84.26, precision:  84.257, recall: 84.257
```

Figure 19: The output data of Model 1 in train data

From the images and data above, we can know that CNN with Batch Normalization but no Dropout has excellent performance in 4 dimensions including Accuracy, Precision and Recall rate and the value of loss function. After 10 epochs in train data, its Accuracy, Precision and Recall rate are all about 84%. After each convolutional layer, there will be a Batch Normalization process, which can reduce the change of the distribution of the inputs of the convolutional layer. It is why model 1 has the performance.

4.3.2 CNN Model 2 (CNN without Batch Normalization and No Dropout)

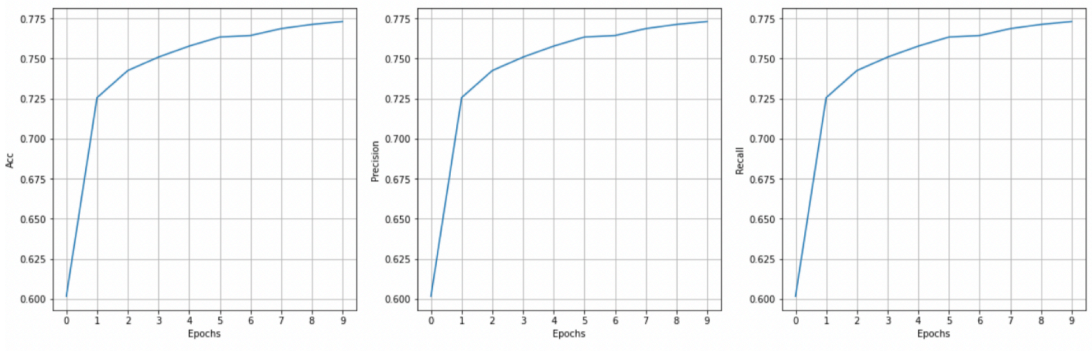

Figure 20: The Accuracy, Precision and Recall rate of Model 2 in train data

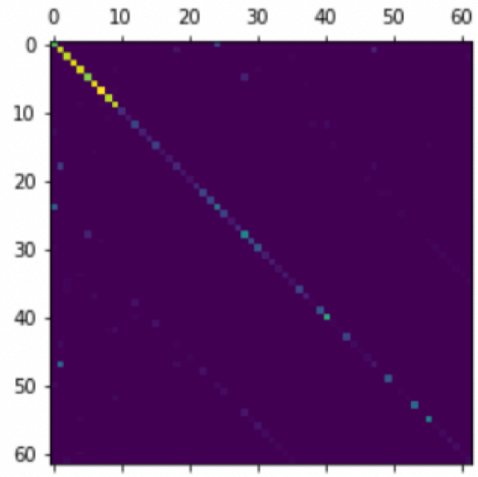

Figure 21: The Confusion matrix Model 2 in train data

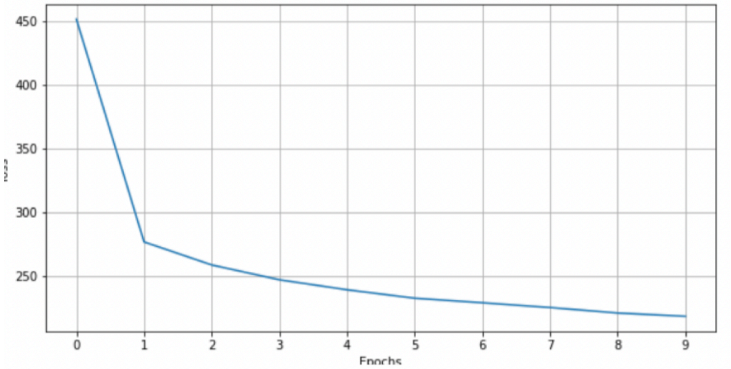

Figure 22: The loss value changes of Model 2 in train data

```
Epoch: 1, loss: 451.54404, Acc: 60.17, precision:  60.167, recall: 60.167
Epoch: 2, loss: 276.95720, Acc: 72.55, precision:  72.552, recall: 72.552
Epoch: 3, loss: 258.90633, Acc: 74.24, precision:  74.243, recall: 74.243
Epoch: 4, loss: 247.30165, Acc: 75.08, precision:  75.080, recall: 75.080
Epoch: 5, loss: 239.44159, Acc: 75.77, precision:  75.767, recall: 75.767
Epoch: 6, loss: 232.83919, Acc: 76.33, precision:  76.333, recall: 76.333
Epoch: 7, loss: 229.28354, Acc: 76.43, precision:  76.428, recall: 76.428
Epoch: 8, loss: 225.54434, Acc: 76.85, precision:  76.855, recall: 76.855
Epoch: 9, loss: 221.23562, Acc: 77.12, precision:  77.120, recall: 77.120
Epoch: 10, loss: 218.64959, Acc: 77.30, precision:  77.300, recall: 77.300
```

Figure 23: The output data of Model 2 in train data

From the images and data above, we can know that CNN without Batch Normalization and Dropout has worse performance then CNN Model 1 after 10 epochs. In 4 dimensions including Accuracy, Precision and Recall rate, the data of model 2 is just about 77%. Its value of loss function is close to 220, which is much higher than model 1. We think that there might be large input values before one of the convolutional layers in it, which is too big too influences the distribution of the original data. Then the generalization of the model is influenced too so CNN model 2 has worse performance than CNN model 1.

4.3.3 CNN Model 3 (CNN with Batch Normalization and Dropout)

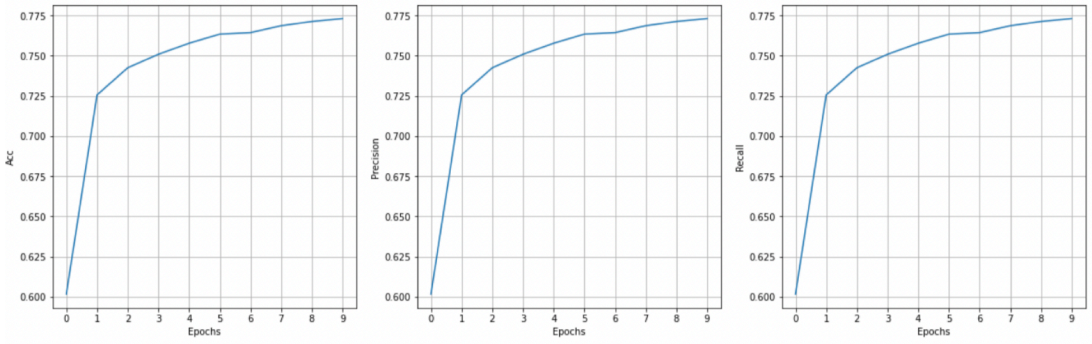

Figure 24: The Accuracy, Precision and Recall rate of Model 3 in train data

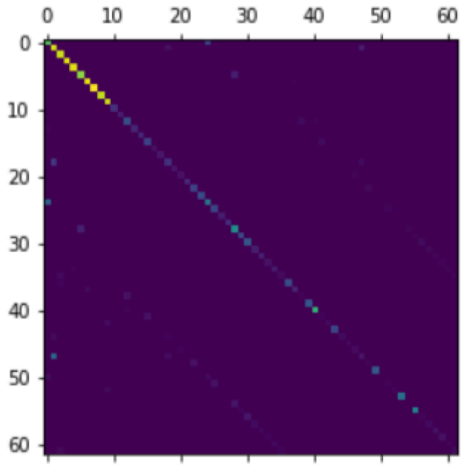

Figure 25: The Confusion matrix Model 3 in train data

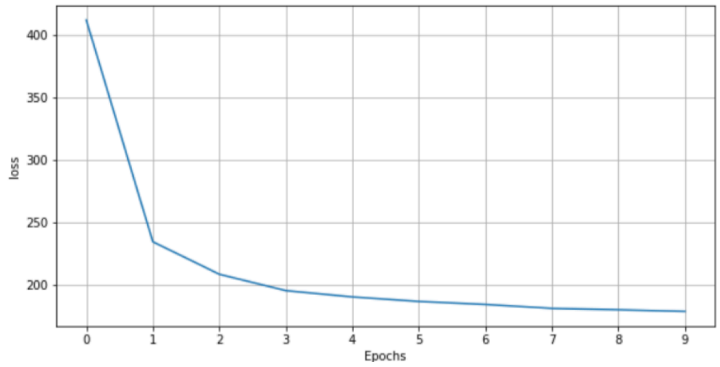

Figure 26: The loss value changes of Model 3 in train data

```
Epoch: 1, loss: 412.02949, Acc: 64.89, precision:  64.885, recall: 64.885
Epoch: 2, loss: 234.32929, Acc: 76.11, precision:  76.110, recall: 76.110
Epoch: 3, loss: 208.29136, Acc: 78.26, precision:  78.263, recall: 78.263
Epoch: 4, loss: 195.15224, Acc: 79.42, precision:  79.420, recall: 79.420
Epoch: 5, loss: 190.12226, Acc: 79.67, precision:  79.668, recall: 79.668
Epoch: 6, loss: 186.48560, Acc: 80.14, precision:  80.143, recall: 80.143
Epoch: 7, loss: 184.08958, Acc: 80.30, precision:  80.300, recall: 80.300
Epoch: 8, loss: 180.95876, Acc: 80.69, precision:  80.692, recall: 80.692
Epoch: 9, loss: 179.81414, Acc: 80.83, precision:  80.825, recall: 80.825
Epoch: 10, loss: 178.49552, Acc: 80.83, precision:  80.828, recall: 80.828
```

Figure 27: The output data of Model 3 in train data

In images and data before, we can see that Accuracy, Precision and Recall rate of CNN model 3 are all about 80%, and the value of loss function is about 178, which means the performance of CNN with Batch Normalization and Dropout (CNN Model 3) is between model1 and CNN model3 in train data. After our analysis we find that the although the dropout technology can reduce the degree of overfitting of model in training process, it might loss some important data. Besides, since the effect of some neuron nodes is randomly eliminated at each iteration, it is not possible to ensure which parameters are mode important in whole parameters. It means that is harder to find best parameters in the model.

After the previous processes, we fined CNN model 1 (Convolutional Neural Network with Batch Normalization but No Dropout) has the best performance in train dataset. CNN model 1 is the best model and it will be used in test dataset. The output of the CNN model for the best model in the test dataset is shown below.

```
Acc: 82.06%, precision:  82.060, recall: 82.060
```

Figure 28: The output of CNN model 1 in test dataset

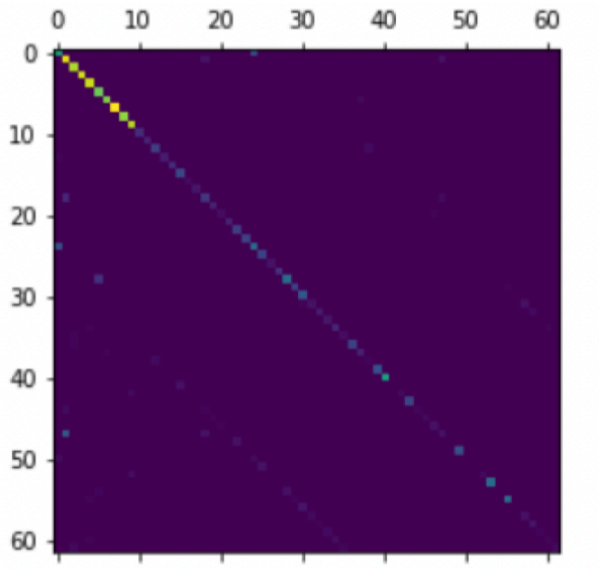

Figure 29: The confusion matrix of CNN model 1 in test dataset

4.4 Evaluation indicators

After performing the three three classification algorithms mentioned above, we obtained different accuracy, precision and recall rates, which we compared specifically, and the results are shown in the table below.

Table 2 Evaluation index table of three classification algorithms

	Accuracy	**Precision**	**Recall**
SVM	0.2926	0.293	0.293
Random Forest	0.3685	0.368	0.368
CNN	0.8206	0.82	0.82

From the table we can clearly see that the CNN performs better than the other two algorithms in terms of accuracy, precision and recall, with the SVM and Random Forest performing closer to each other. To further compare the three algorithms, we also compared the confusion matrices of the three algorithms and the results are shown in the figure below.

5 Conclusion and Future Work

5.1 Conclusion

In this experiment, we analysed and evaluated the performance of three different classification algorithms on the EMNIST ByClass dataset. From the experimental results, the CNN

model showed the best performance on the test set with an accuracy of about 82%, while the SVM and Random Forest performed relatively poorly with an accuracy of 29.26% and 36.85% respectively. On the other hand, the CNN has a longer overall running time, computes a larger amount of data and consumes more computer resources as it requires many cycles of training, however, considering that CNN can extract local features and has a better classification effect on images, we finally chose CNN as our best model.

5.2 Reflection

When conducting CNN classification experiments, we can clearly feel the impact of different parameter settings on the experimental results. The large amount of computation cannot be trained quickly and smoothly on our weak computer hardware, which shows that there is still much room for improvement and optimisation in the architecture of the algorithm.

5.3 Future Work

In a previous study, researchers combined AdaBoost with a convolutional neural network (CNN), an algorithm that reduces the computational cost when processing large data sets by reducing the learning period of the component estimator, which in turn improves the training progress and reduces the training time. Therefore, in subsequent research, we can use a complementary approach to combine different algorithms based on the implementation principles of the different algorithms to achieve better experimental results, depending on the actual characteristics of the data.

6 APPENDIX

6.1 Hardware and Software Specifications

The code was created on the Anaconda platform using a jupyter laptop. The laptop we used is an Apple Macbook Pro, which has the following configuration.

- Processor: Apple M1 Pro
- OS: macOS Monterey
- RAM: 16 GB
- Model: MacBook Pro (14inch, 2021)

6.2 Instructions for running the code

The following is a description of the code file to run for this project.

1. LOAD DATA Loading data is done differently in different environments. You can run our code by downloading the dataset through Colab, as shown below, minus all the symbols #, or you can download the dataset and run it locally.

```
# from pydrive.auth import GoogleAuth
# from pydrive.drive import GoogleDrive
# from google.colab import auth
# from oauth2client.client import GoogleCredentials

# auth.authenticate_user()
# gauth =GoogleAuth()
# gauth.credentials =GoogleCredentials.get_application_default()
# drive =GoogleDrive(gauth)

# file_id ='1_J2x82HkJJX2p-EfGRjQZk2-aENG-Z6z'
# downloaded = drive.CreateFile({'id':file_id})
# downloaded.GetContentFile("emnist-byclass.mat")
```

Figure 30: Schematic diagram of importing data

2. IMPLEMENTATION

   We used 3 different classification algorithms, divided into 3 different ipynb files, you can choose one of them to run. Among them CNN we used 3 different ways to train, model 1 is the best model.

2. OUTPUT

   Finally, we output the final prediction model, which is saved locally and the best model is selected for uploading to Canvas.

References

[1] G. Biau and E. Scornet. A random forest guided tour. volume 25, pages 197–227. Springer, 2016.

[2] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. pages 2921–2926, 2017.

[3] J. A. Deatrick, K. A. Knafl, and C. Murphy-Moore. Clarifying the concept of normalization. volume 31, pages 209–214. Wiley Online Library, 1999.

[4] S. Hijazi, R. Kumar, C. Rowen, et al. Using convolutional neural networks for image recognition. volume 9, 2015.

[5] A. Maćkiewicz and W. Ratajczak. Principal components analysis (pca). volume 19, pages 303–342. Elsevier, 1993.

[6] W. S. Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[7] P. Probst, M. N. Wright, and A.-L. Boulesteix. Hyperparameters and tuning strategies for random forest. volume 9, page e1301. Wiley Online Library, 2019.

[8] A. Taherkhani, G. Cosma, and T. M. McGinnity. Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, 404:351–366, 2020.

[9] S. Wang and C. Manning. Fast dropout training. In *international conference on machine learning*, pages 118–126. PMLR, 2013.