Fashion Data Classification

Marjan Emadi
memadi@eng.ucsd.edu

Zixin Ma
zim023@ucsd.edu

Yijing Li
yil232@eng.ucsd.edu

Runze Xu
r1xu@ucsd.edu

Zicong Zhang
ziz059@ucsd.edu

June 15, 2019

Abstract

The link to our github repository is https://github.com/wkcw/ece228 (Note: Because we do not know if a public github repo violates Academic Integrity, the repo is private. We posted a question on piazza. Please tell us if we should make it public. We will modify it at once.). The objective of this project is to utilize different machine learning models to classify fashion item images in the Fashion-MNIST dataset into certain categories. We trained seven different models including convolutional neural network (CNN), ResNet, VGG, and other conventional models. In this paper, we will explain each method we used and compare the accuracy resulted from them.

Figure 1: Examples from Fashion-MNIST dataset

1 Introduction

Classification is one of the main problems in machine learning. In this project, a fashion-MNIST dataset, which includes figures of ten different fashion classes, is classified through various machine learning methods using TensorFlow, Keras and scikit-learn.

Nowadays, there are numerous images of clothing and outfits online that could suggest potential business opportunities. For example, by categorizing existing clothing and outfits of customers from different age groups, we can predict different people's preferences and build recommending system for clothing to targeting customers. Therefore, an accurate and efficient method to search through images and classify them into different categories can be extremely important.

The input to our algorithm is a greyscale fashion image, for example, shoes, t-shirt, dress, etc.. We then use different machine learning models, such as CNN, ResNet, VGG, random forest, KNN, SVM, and Adaboost to output a predicted class for it.

2 Related Work

Resnet [He et al., 2016] is proposed to solve the common issues in training deep neural networks, Gradient Vanishing and curse of dimensionality problems. The model has achieved great success on image classification problems. This model makes training deep models much easier than before. It works especially well when the input data is not so complicated while the network is big. Resnet model has many different implementations from 18 layers to 152 layers. The number of layers is extremely large when Resnet was proposed. It successfully trained the 152-layer model with its residual block design.

As for the conventional machine learning methods, we directly applied Scikit-learn modules in python[Pedregosa et al., 2011]. The detailed algorithms are explained in detail in section 5.3.

3 Dataset and Features

The dataset we used for our project is the Fashion-MNIST dataset provided by Kaggle. Within our dataset, we have 10 different categories and 70,000 unique images, including a training set of 60,000 images and a test set of 10,000 images. Each of the image is a 28 by 28 greyscale image. We also tried standarding the data to make it between a minimum value of 0 and a maximum value of 1. But this pre-processing step did not improve the result. We think the reason is that we used batch normalization in our models, which already made use of standardizing data.

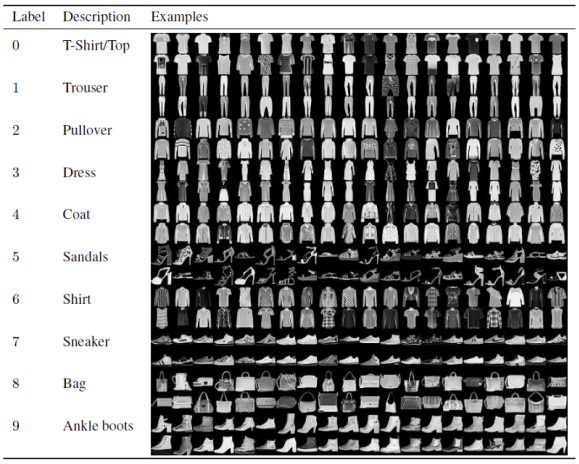


Figure 2: Ten different classes and corresponding samples from our dataset

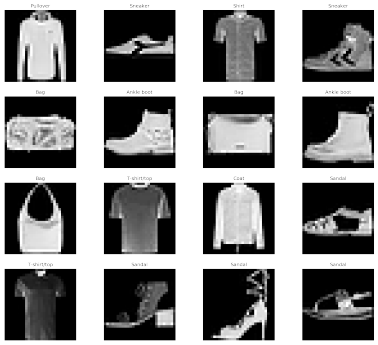


Figure 3: Plots of some random samples from our dataset with corresponding labels

4 Methods

4.1 Convolutinal Neural Network (CNN)

CNN [Simard et al., 2003] has been proved to be the most successful image classifier. Its capability to extract features from input regardless of space information makes it immune to the position of a key object in the image.

The convolutional neural network uses a series of filter layers to constantly extract useful features and reduce dimensions on raw data, the resulting data will then be evaluated for final classification to certain node (category).

4.2 ResNet

Our next model is Resnet. Resnet has achieved huge success in image classification tasks. The design of this model is driven by a simple intuition. The bigger (more layers and more parameters) the model is, the better it should be. However, in practice we often witness the fact that complicated networks generate poor results than simple networks. The reason lies in that weights in early layers of the model is not well updated by the back-propagated gradients. Resnet addresses this issue by adding a shortcut at between certain layers. Because our task is to classify 28x28x1 images which are relatively small and we found VGG works worse than a simple and shallow CNN, we want to see if Resnet can solve the problem. In the Section 4, we introduced our result and gave our analysis on Resnet.

4.3 VGG

In this experiment, we used VGG16[1] as a method for classifying our data. VGG16 is a deep and simple network, that uses 16 convolution filters with the kernel size of 3 for all, and 5 maxpooling for increasing the effective receptive field. When the convolutions and maxpooling is done, it appends 3 fully connected layers as classifiers, to detect the class of the input data. VGG16 gives better results compared to some networks like ZFNet. Since it is using more convolution layers, its effective receptive field is bigger, which means it can extract more feature from the data. This also means that it uses more activation functions,that causes VGG16 to be more discriminant.

4.4 Conventional Machine Learning Methods

Apart from CNN, ResNet and VGG, some conventional machine learning methods are also applied in this classification problem, including Random Forest, k-Nearest Neighbor (kNN), Support Vector Machine (SVM) and Adaboost.

4.4.1 Random Forest

Random Forest [Breiman, 2001] is an extension of bagging algorithm [Breiman, 1996] with decision trees as base learners. Compared to bagging, random forest is simpler in implementation, and enjoys lower computation cost. Traditional decision tree choose the best representing feature among the union of all the features of current node. However, in random forest, a subset including k features will be chosen randomly, from which the best representing feature will be selected. Therefore, random forest usually shows higher efficiency than traditional bagging algorithm. As for the convergence, random forest algorithm acts similarly as bagging. However, random forest usually has worse performance in starting stages, especially when there is only one base learner. When the number of base learners increase, usually random forest will converge to get lower errors than bagging.

4.4.2 k-Nearest Neighbor

k-Nearest Neighbor (kNN) learning is one of the commonly used methods in supervised learning. The mechanism can be easily understood. For a given testing set, k nearest training samples will be selected, and predictions will be based on these k "neighbors" by voting or averaging in general. k-Nearest Neighbor is also known as a method of lazy learning, which has no running time cost in the training stage. It saves the training samples and processes them in the later stage. Although kNN algorithm is simple, the risk of kNN is still less than 2 times of optimal Bayes risk.[Cover et al., 1967] While kNN enjoys its strength is low cost, usually the accuracy of results of kNN is lower than that of CNN.

4.4.3 SVM

Support Vector Machine [Suykens and Vandewalle, 1999] is a type of supervised learning which classifies the dataset through linear classification. It is usually realized by finding a hyper-plane between 2 classes.

4.4.4 Adaptive Boosting

Boosting is an algorithm which improves weak learner to strong learners. In boosting, a base learner will firstly be generated from the original training set. After that, the weighting of samples will be adjusted accordingly, addressing more attention to the wrong predictions in training set of base learners. Based on the adjustments, the latter base learner will be trained. These steps will be repeatedly processed until the last stage, where all the base learners are combined by certain weighting.

Adaptive boosting introduced by Freund and Schapire[Freund and Schapire, 1997], which is also know as Adaboost, is one of the most famous algorithms in boosting. It uses additive model, in other words, the linear combination of base learners, to minimize the exponential loss function[Friedman et al., 2000].

5 Experiments and Results

In this section, we will introduce our experiment results and analysis for the above methods mentioned.

5.1 Convolutional Neural Network

5.1.1 Model Structure and Layers Order

Because we are new to Deep Learning, the implementation of our CNN model used the code provided in the official tutorial of Tensorflow. The model is assembled with several layers in the following order.

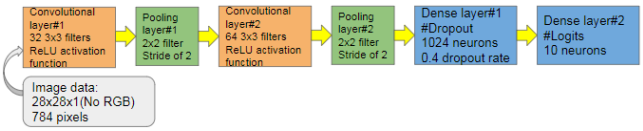


Figure 4: Model structure

The **convolutional** layers are used to extract useful subregions (features), and we are allowing padding here to preserve as much information as we can. Each convolutional layer is followed by an activation function to introduce nonlinearities to the model. We choose to use ReLU(Rectified Linear Units) function as our activation function to make the training faster and fix gradient problem (compared to other nonlinear functions)

The **pooling** layers are used to downsample the extracted data after each convolutional layer. For our approach, we are using a 2x2 filter with stride set to 2, and we are using the max pooling as our pooling strategy.

Eventually, the extracted data will be evaluated by two consecutive **dense layers** called **Dropout layer** and **Logits layer**. Both dropout layer and logits layer do the classification based on preceding layers. However, the dropout layer also drops part of activation (data) to avoid overfitting, so the model becomes less used to predicting training data and increase its performance

when predicting based on test data. Eventually, the logits layer will give a possibility for each node (category) and determine which category best describes this item.

5.1.2 Improvement and Results

The best accuracy we achieved with CNN is 0.897. To increase the model's the accuracy at predicting and decrease loss, we changed several parameters inside the model for enhanced performance.

Convolutional layers: By comparing performance under different filter size, we found that 6x6 filter gives the best accuracy.

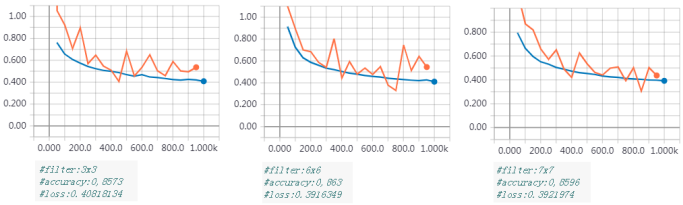


Figure 5: Performance difference under different filter size

Dropout layers: The dropout rate determines how likely we will drop activation, either too low (overfitting) or too high (unable to predict at all) will decrease the model performance. In our experiment, 0.2 will promise the best performance after our experiments. (the performance difference between 0.2 and other dropping rates is limited, but 0.2 is still the best choice)

Training steps: we set the training steps to be high enough for maximum performance and decent training time required. In our experiments, the model reaches its maximum performance with 10000 steps.

With the optimization on all parameters above, the accuracy can reach 0.95 for maximum.

5.2 ResNet

Our Resnet experiment is designed to reveal the influence of different model structure and hyper-parameters on the final result. Concretely, we evaluate the influence of Training Optimizer, Number of Channels in layers and most importantly the position of shortcut in model. To simplify our description, we call the by-pass in a residual block of Resnet "shortcut", while we call the rest part of a residual block "main layers". Out of hardware concern, we implemented the Resnet-18 model as our base model. Bigger resnet

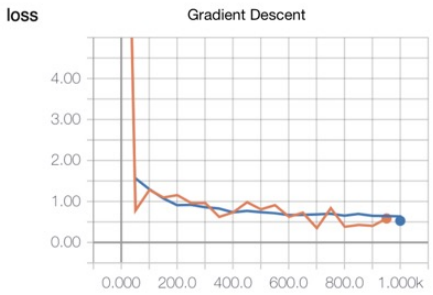


Figure 6: GradientDescent train loss and test loss curve.

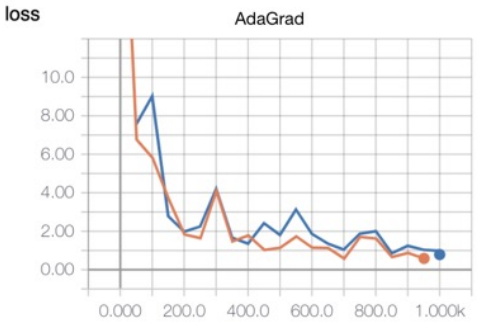


Figure 7: AdaGrad train loss and test loss curve.

model causes Out of Memory Error when we train the model on DataHub.

5.2.1 Training Optimizer

The ability of different Training Optimizer is well-studied. The purpose of doing this experiment is to find the best optimizers for latter experiments. For our task, we have applied Gradient Descent, Adam, Towers, AdaGrad and RMSprop optimizer. Each of these optimizers are tuned with Grid Search on the learning rate and some other hyper-parameters. The experiment result shows that AdaGrad achieves the best test set accuracy. Partial loss curve is shown in Figure 6 and Figure 7:

For the rest of our experiment on Resnet, we all adopted AdaGrad Optimizer with hyper-parameters here.

5.2.2 Number of Weights

Models with more weights do not necessarily may lead to worse result than models with less weights because of Gradient Vanishing. The residual block in Resnet is designed to address this issue. After studying Resnet, however, we found a common wrong understanding of Resnet is that the shortcut in residual block

	original size	1/2 parameters	1/4 parameters	1/8 parameters	1/16 parameters
loss/ accuracy	0.62/83.5%	0.43/87.4%	0.37/87.3%	0.38/87.3%	0.40/86.8%

Figure 8: Result of different model size choices.

	All Shortcuts	only 4th shortcut	only 3rd shortcut	only 2nd shortcut	only 1st shortcut
loss/ accuracy	0.42/85.4%	0.47/84.3%	0.51/83.1%	0.49/83.8%	0.63/77.8%

Figure 9: Result of different shortcut position.

has no weights so the shortcut ensures the gradient to be effectively back-propagated to the first few layers. In fact, the shortcut is a CNN layer with 1x1 kernel. It expands the channel number of input to the channel number of the ouput from main layers of the residual block, usually twice the input channel number. Therefore, the shortcut cannot completely eliminate Gradient Vanishing. And tuning the number of Weights is still indispensable. The result in Figure 17 shows that the best result is achieved when the number of weights in half of the original.

5.2.3 Shortcut Placement

As the core of Resnet, shortcut is placed at every block (two layers as one block) except the first one. We evaluated the effect of shortcut by retaining only one of those shortcuts. The only shortcut is placed at different layers to see which residual is the most crucial one. We also compared the one-shortcut model with original all-shortcut model. The result is shown in Figure 9.

We can see the deeper you place the shortcut, the better the result. We think this is related to the property of gradient back-propagation. Shortcut helps relieve Gradient Vanishing, but if it is placed in shallow layers, the gradient is already very small before reaching this layer. Moreover, all-shortcut model outperforms all the others, which is reasonable since it keeps the gradient effective to the greatest extent.

5.3 Conventional Methods Results

5.3.1 Results

Random Forest Classifier The validation curve which shows the correlation between accuracy score and number of trees is shown below. It is observed that for this particular fashion classification problem, the best amount of trees is around 10.

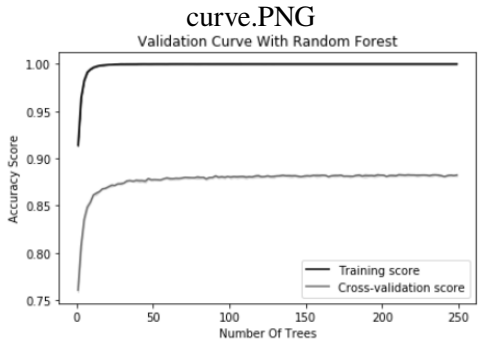


Figure 10: Validation Curve of Random Forest Classifier

Therefore for this classification problem, the number of trees in the forest is set to be 10, and $k = \sqrt{(n_{features})}$. The accuracy score for the classification is 0.8617. We also tried $k = log_2(n_{features})$ and obtained an accuracy score of 0.8509 which is slightly lower than the previous setting.

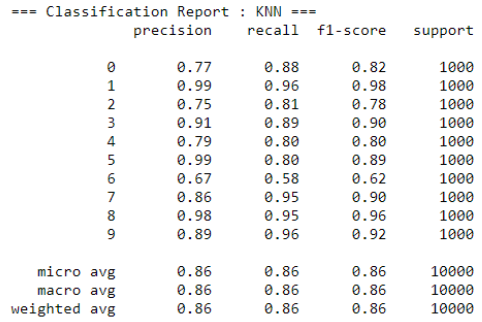
```
=== Classification Report : Random Forest ===
              precision    recall  f1-score   support

           0       0.76      0.84      0.80      1000
           1       0.98      0.97      0.97      1000
           2       0.75      0.78      0.77      1000
           3       0.88      0.90      0.89      1000
           4       0.77      0.79      0.78      1000
           5       0.95      0.94      0.95      1000
           6       0.69      0.55      0.61      1000
           7       0.91      0.91      0.91      1000
           8       0.95      0.96      0.96      1000
           9       0.93      0.94      0.94      1000

   micro avg       0.86      0.86      0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000
```

Figure 11: Classification Report for Random Forest Classifier

k-Nearest Neighbor For the given classification problem, k is set to be 10. The running time of kNN is relative short compared to all other methods we have mentioned in this report. The accuracy of classification for kNN is around 0.8573.

Support Vector Machine In this paper, linear support vector classification (SVC) in sklearn is applied. Gamma is set as "scale", which means we use $1/(n_{features} \times X.var())$ as values of gamma. Although the running time of SVM is much longer than that of the methods mentioned above, the accuracy score is a bit higher, which is around 0.89.

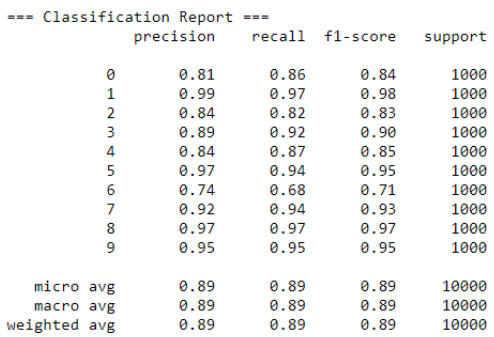
Adaptive Boosting For this classification problem, previously we tried to use the default decision tree classifier as the base learner. With $n_{estimators}$ set to be



=== Classification Report : KNN ===

	precision	recall	f1-score	support
0	0.77	0.88	0.82	1000
1	0.99	0.96	0.98	1000
2	0.75	0.81	0.78	1000
3	0.91	0.89	0.90	1000
4	0.79	0.80	0.80	1000
5	0.99	0.80	0.89	1000
6	0.67	0.58	0.62	1000
7	0.86	0.95	0.90	1000
8	0.98	0.95	0.96	1000
9	0.89	0.96	0.92	1000
micro avg	0.86	0.86	0.86	10000
macro avg	0.86	0.86	0.86	10000
weighted avg	0.86	0.86	0.86	10000

Figure 12: kNN Classification report

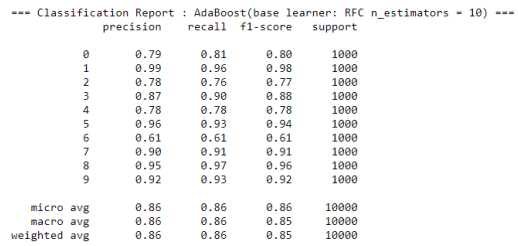


=== Classification Report ===

	precision	recall	f1-score	support
0	0.81	0.86	0.84	1000
1	0.99	0.97	0.98	1000
2	0.84	0.82	0.83	1000
3	0.89	0.92	0.90	1000
4	0.84	0.87	0.85	1000
5	0.97	0.94	0.95	1000
6	0.74	0.68	0.71	1000
7	0.92	0.94	0.93	1000
8	0.97	0.97	0.97	1000
9	0.95	0.95	0.95	1000
micro avg	0.89	0.89	0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

Figure 13: SVC Classification Report

50 and algorithm set to be $SAMME.R$, we only got an accuracy score of approximately 0.58 when using the default base learner. However, when Random Forest Classifier is applied (with $n_{estimators} = 10$), the accuracy score increased dramatically to 0.8654, which is slightly better than the results of pure Random Forest Classifier.



=== Classification Report : AdaBoost(base learner: RFC n_estimators = 10) ===

	precision	recall	f1-score	support
0	0.79	0.81	0.80	1000
1	0.99	0.96	0.98	1000
2	0.78	0.76	0.77	1000
3	0.87	0.90	0.88	1000
4	0.78	0.78	0.78	1000
5	0.96	0.93	0.94	1000
6	0.61	0.61	0.61	1000
7	0.90	0.91	0.91	1000
8	0.95	0.97	0.96	1000
9	0.92	0.93	0.92	1000
micro avg	0.86	0.86	0.86	10000
macro avg	0.86	0.86	0.85	10000
weighted avg	0.86	0.86	0.85	10000

Figure 14: AdaBoost Classification Report (Base learner: RFC)

5.3.2 Discussion on traditional methods' results

By comparing the performance of methods above, it is observed that random forest and kNN satisfying when dealing with large dataset. Also, the results of SVM has show higher accuracy than other methods, the reason may be that the correlations are mostly nonlinear, and the variables are almost normally distributed. However, the major downside of SVM is that they can be painfully inefficient, which means they may not be suitable for large dataset training. Boosting usually has better performance than other methods, and the result is consistant with this understanding.

5.4 VGG

We experiment 2 different types of VGG with 16 layers in both.The architecture of the model and training optimizers are going to be discussed in the following subsections.

First we are going to talk about our very basic model used for classifying the data. We call this the VGG basic model. Then, we are going to talk about the second method used to improve the result with using training optimizer, and we are going to discuss how each of them improved the accuracy compared to the basic VGG model.

5.5 Basic VGG16 model

In this model we used VGG16, without much training optimizers.For each epoch, we have minibatches of size 100.for training set of size 60K this means we are going to have 600 updates for weights and biases per epoch, which is a reasonable number. We also used 0.0001 for the learning rate. We did not use a big number since it might cause divergence or smaller one since it might take so long to converge. Also 0.2 of the training set was used as the validation set for early stopping and preventing from overfitting. Using 100 epochs and monitoring validation loss at the same time, the training was stopped at 49th epoch, you can also see this in figure 15. We used accuracy and loss for evaluating the basic VGG model. In the following plots, you can see the loss and accuracy for this model:

5.6 Optimized VGG16 model

In this model, we used more methods for optimizing the model. Since it was taking a long time for training, we increased the learning rate in compare with the base model to 0.001. For each layer we used batch normalization, to help each layer learn more by itself independently from other layers, and reduce over fitting. Since the chance of over fitting has reduced by using batch normalization, we also increased the number of epochs to 200 to learn more features and update them more. We also used dropouts with the value of 0.4, to add more sparsity and improve generalization

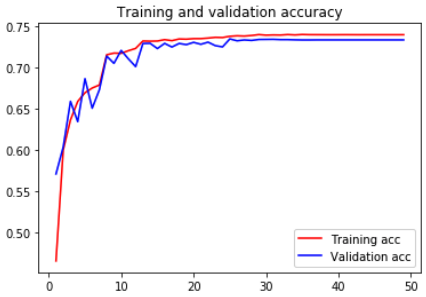


Figure 15: Accuracy result for Basic VGG model during training

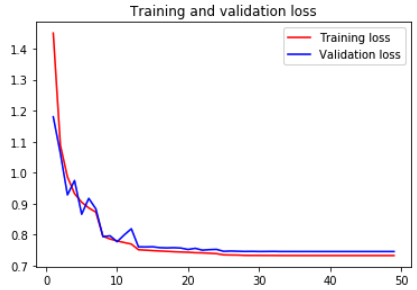


Figure 16: Loss result for Basic VGG model during training

of the model. We also used momentum to improve the regularity of the stochastic gradient descent. By using the mentioned methods above, we were able to increase the accuracy from 0.73 to 0.87.

	Accuracy	Loss
Basic VGG16	0.73	0.73
Optimized VGG 16	0.87	0.75

Figure 17: Accuracy and Loss for 2 models of VGG16

6 Conclusion

We found that although Resnet is claimed to make complicated models perform not worse than simple models, the fact that shortcuts have weights still requires us to tune the model size. The shortcut of Resnet is mostly effective at deep layers of the network.

As for traditional machine learning methods, the accuracy scores are competitive to those of neural networks. However, for SVM, the time cost is much higher. Also, it is observed that Adaboost can help improve the results of other learners. If implemented with CNN, Resnet or VGG, Adaboost may help reach a much satisfied accuracy.

As for CNN and VGG, we also achieved good result. Through careful parameter tuning, the ability of CNN on image classification is shown.

7 Contributions

Marjan Emadi contributed to the parts of VGG method description, experiments and results.

Zixin Ma contributed to the parts of conventional methods' description, experiments and results.

Yijing Li contributed to the abstract, introduction, related work, dataset, and conclusion parts.

Runze Xu contributed to the parts of ResNet method description, experiments and results.

And Zicong Zhang contributed to the parts of CNN method description, experiments and results.

8 References

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45 (1):5–32, 2001.

Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks ap-

plied to visual document analysis. In *Icdar*, volume 3, 2003.

Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.