1. Introduction.

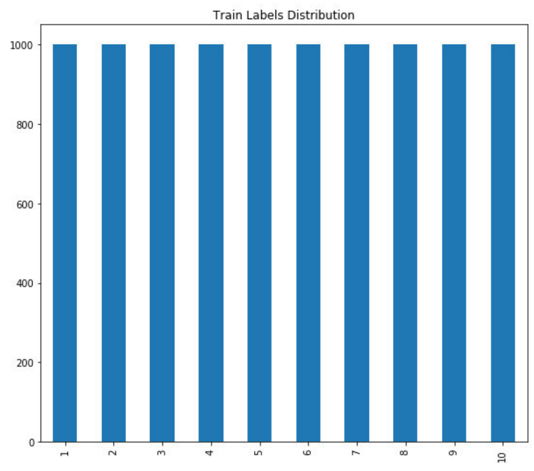
> Introduction. Provide an overview of the problem, your proposed solution, and your experimental results. [10%]

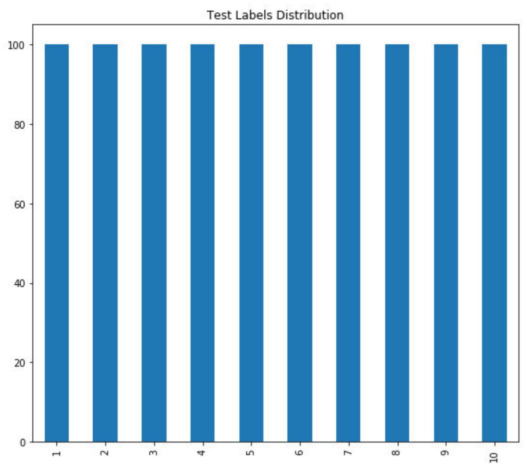
This task is a classification task on the CIFAR-10 subset. In the experiment, I tried to use svm, Kmeans, Gaussian mixture model and neural network model for classification task, and extended the method of transfer learning based on the model in the neural network, and finally obtained the highest accuracy = 0.65 in transfer learning.

2. Method.

> Present your proposed method in detail. This should cover how the features are extracted, any feature processing you use (e.g. clustering and histogram generation, dimensionality reduction), which classifier(s) is/are used, and how they are trained and tested. This section may contain multiple sub-sections. [50%]

- 2.1 First, read the original data and study the distribution of the labels. The distribution results are as follows.





- 2.2 Secondly use the `computeFeatures` method provided in the `RunMe.ipynb` file to preprocess the original picture matrix `trnImages,` `tstImages`, and obtain the hog feature vector `trnFeatures, tstFeatures` of each picture. For comparison, `sklearn.preprocessing.StandardScaler` is also used to normalize the extracted hog feature vector to obtain the standardized hog feature vector `trnFeatures_scale, tstFeatures_scale`.

- 2.3 In the modeling part, the four classification models of svm, Kmeans, Gaussian Mixture Model and Neural Networks are used for training and testing on two feature vectors `trnFeatures, tstFeatures`, and `trnFeatures_scale, tstFeatures_scale`. For the test results, see part 3. Here Neural Networks is implemented using `sklearn.neural_network.MLPClassifier`. The parameters are all defaulted, as shown in the following table.

parameter	value
learning_rate_init	0.001
hidden_layer_sizes	(100,)
activation	relu
solver	adam

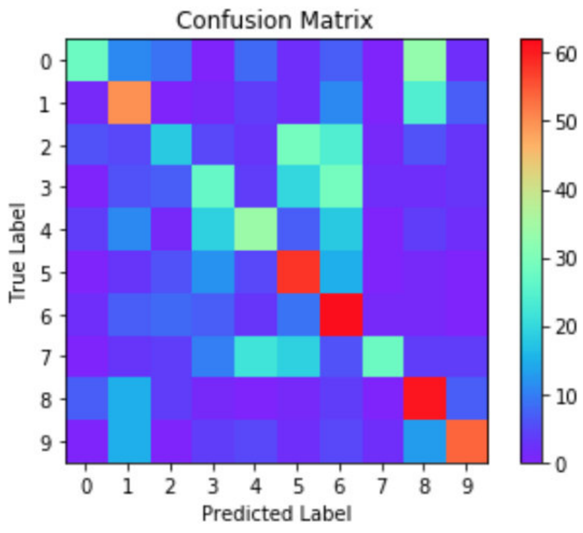
- 2.4 Because the effect of Neural Networks is greatly affected by the model structure, and considering that `hog feature vector` is not the best feature, using CNN structure can automatically learn the features of the image. In addition, in the process of consulting the information, I learned the concept of Transfer Learning. In practice, few people train entire convolutional networks from scratch (random initialization) because it is relatively rare to have a sufficiently large data set. Instead, it is common practice to pre-train ConvNet on very large data sets (such as ImageNet, which contains 1.2 million images with 1000 classes), and then use ConvNet to initialize tasks of interest or use them as fixed feature extractors . Therefore, I have extended a method based on Transfer Learning based on the Neural Networks model. The specific method is to use the Resnet18 model pre-trained on the ImageNet full data set as the basic model, and use the data in this task to train the model. During training, only the parameters of the last fully connected layer are adjusted.

3. Results.

> Present your experimental results in this section. Explain the evaluation metric(s) you use and present the quantitative results (including the confusion matrix). If you have tried multiple solutions, present all the results. [30%]

When evaluating the classification model performance on the test set, you can generally use the parameters accuracy / precision / recall / f1-score. When evaluating the performance of the model, the visualization `confusion matrix` and `sklearn.metrics.classification_report` are mainly used. Way to embody. The experimental results are shown below.

- 3.1.a SVM + HOG feature

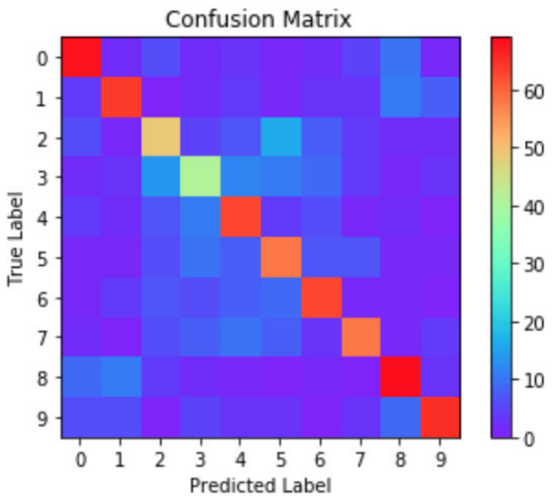


```
              precision    recall  f1-score   support

           1       0.28      0.58      0.38        48
           2       0.50      0.40      0.44       126
           3       0.18      0.32      0.23        57
           4       0.27      0.31      0.29        86
           5       0.34      0.39      0.36        88
           6       0.58      0.39      0.47       149
           7       0.62      0.34      0.44       181
           8       0.28      0.82      0.42        34
           9       0.61      0.41      0.49       149
          10       0.54      0.66      0.59        82

    accuracy                           0.42      1000
   macro avg       0.42      0.46      0.41      1000
weighted avg       0.48      0.42      0.43      1000
```

- 3.1.b SVM + standarlize HOG feature

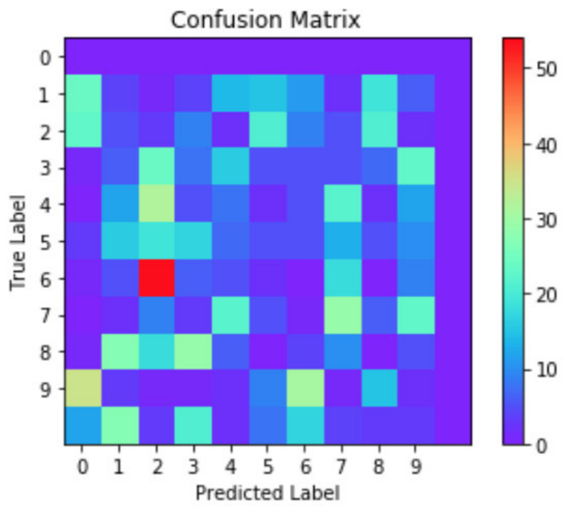


```
              precision    recall  f1-score   support

           1       0.68      0.66      0.67       103
           2       0.64      0.68      0.66        94
           3       0.49      0.49      0.49        99
           4       0.41      0.45      0.43        92
           5       0.63      0.53      0.58       119
           6       0.58      0.52      0.55       111
           7       0.63      0.62      0.62       102
           8       0.58      0.67      0.62        86
           9       0.69      0.64      0.67       107
          10       0.65      0.75      0.70        87

    accuracy                           0.60      1000
   macro avg       0.60      0.60      0.60      1000
weighted avg       0.60      0.60      0.60      1000
```

- 3.2.a Kmeans + HOG feature

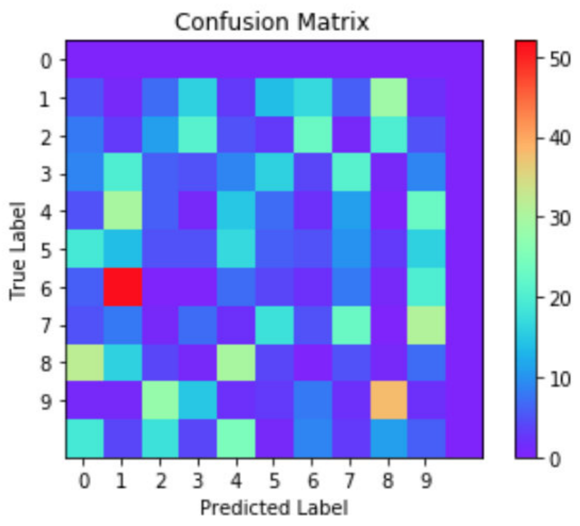


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       100
           1       0.04      0.04      0.04       107
           2       0.03      0.02      0.02       164
           3       0.08      0.08      0.08       103
           4       0.08      0.10      0.09        84
           5       0.05      0.07      0.06        72
           6       0.00      0.00      0.00        88
           7       0.29      0.27      0.28       109
           8       0.00      0.00      0.00        78
           9       0.02      0.02      0.02        95
          10       0.00      0.00      0.00         0

    accuracy                           0.06      1000
   macro avg       0.05      0.05      0.05      1000
weighted avg       0.06      0.06      0.06      1000
```

- 3.2.b Kmeans + standarlize HOG feature

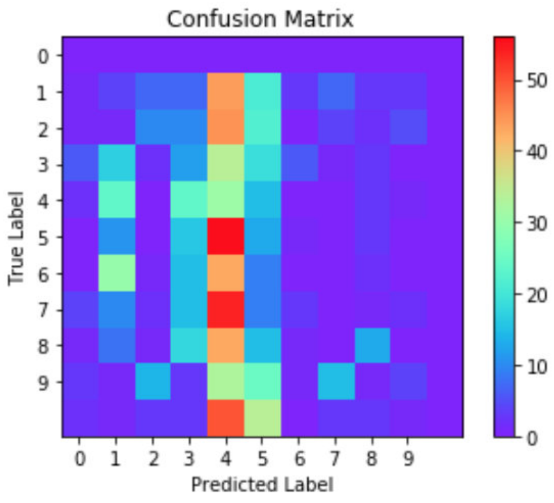


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       109
           1       0.01      0.01      0.01       149
           2       0.11      0.13      0.12        86
           3       0.05      0.07      0.06        75
           4       0.15      0.13      0.14       115
           5       0.06      0.08      0.07        76
           6       0.02      0.03      0.02        75
           7       0.23      0.26      0.24        90
           8       0.01      0.01      0.01       104
           9       0.02      0.02      0.02       121
          10       0.00      0.00      0.00         0

    accuracy                           0.07      1000
   macro avg       0.06      0.07      0.06      1000
weighted avg       0.06      0.07      0.06      1000
```

- 3.3.a Gaussian Mixture Model + HOG feature

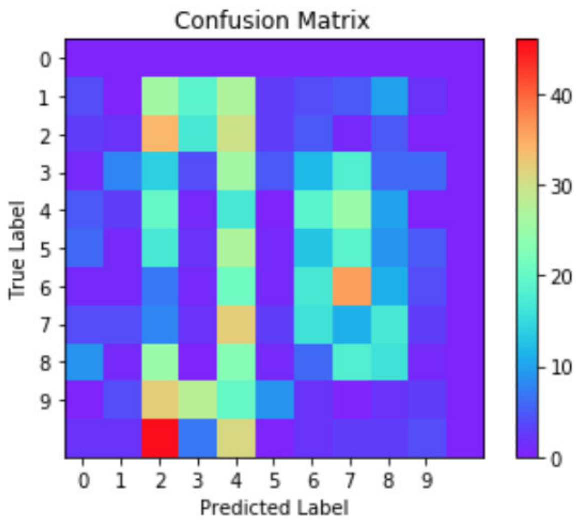


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        20
           1       0.04      0.04      0.04       107
           2       0.10      0.25      0.14        40
           3       0.12      0.10      0.11       123
           4       0.31      0.07      0.12       433
           5       0.13      0.07      0.09       182
           6       0.00      0.00      0.00        15
           7       0.00      0.00      0.00        30
           8       0.13      0.38      0.19        34
           9       0.04      0.25      0.07        16
          10       0.00      0.00      0.00         0

    accuracy                           0.09      1000
   macro avg       0.08      0.11      0.07      1000
weighted avg       0.19      0.09      0.10      1000
```

- 3.3.b Gaussian Mixture Model + standarlize HOG feature

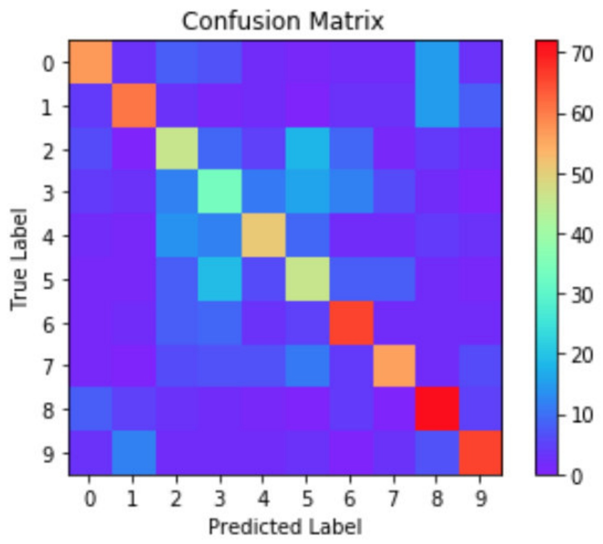


```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        35
           1       0.00      0.00      0.00        26
           2       0.34      0.15      0.21       229
           3       0.04      0.05      0.04        81
           4       0.17      0.07      0.10       254
           5       0.01      0.04      0.02        26
           6       0.17      0.18      0.17        96
           7       0.11      0.08      0.09       136
           8       0.16      0.18      0.17        89
           9       0.03      0.11      0.05        28
          10       0.00      0.00      0.00         0

    accuracy                           0.10      1000
   macro avg       0.09      0.08      0.08      1000
weighted avg       0.17      0.10      0.12      1000
```

- 3.4.a Neural Networks + HOG feature

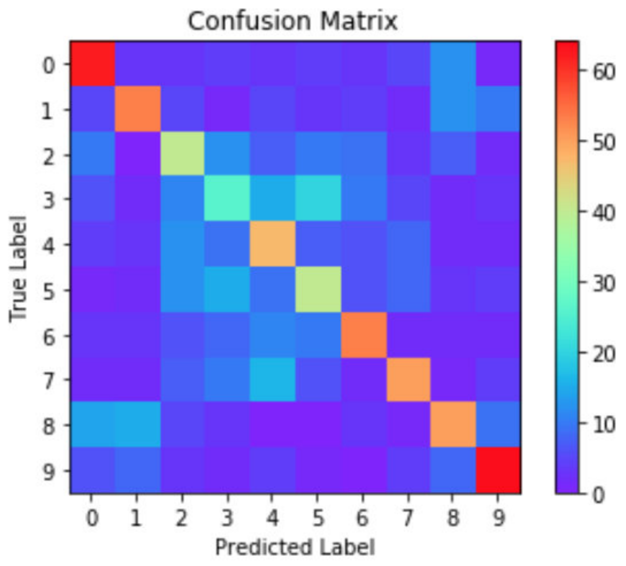


```
              precision    recall  f1-score   support

           1       0.57      0.66      0.61        87
           2       0.61      0.69      0.65        88
           3       0.46      0.42      0.44       110
           4       0.34      0.33      0.34       102
           5       0.51      0.57      0.54        90
           6       0.46      0.42      0.44       109
           7       0.66      0.60      0.63       110
           8       0.56      0.67      0.61        83
           9       0.72      0.58      0.64       125
          10       0.66      0.69      0.67        96

    accuracy                           0.56      1000
   macro avg       0.55      0.56      0.56      1000
weighted avg       0.56      0.56      0.55      1000
```

- 3.4.b Neural Networks + standarlize HOG feature

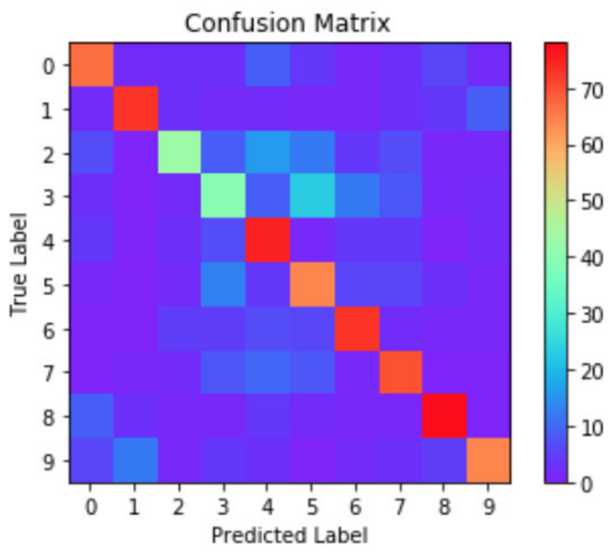


```
              precision    recall  f1-score   support

           1       0.62      0.55      0.58       113
           2       0.53      0.58      0.55        91
           3       0.40      0.38      0.39       104
           4       0.26      0.29      0.27        90
           5       0.47      0.40      0.43       117
           6       0.40      0.40      0.40       101
           7       0.53      0.55      0.54        96
           8       0.50      0.57      0.53        88
           9       0.50      0.51      0.50        99
          10       0.64      0.63      0.64       101

    accuracy                           0.48      1000
   macro avg       0.48      0.49      0.48      1000
weighted avg       0.49      0.48      0.49      1000
```

- 3.5 Neural Networks + Transfer Learning



```
              precision    recall  f1-score   support

           0       0.68      0.67      0.67       100
           1       0.80      0.73      0.76       100
           2       0.66      0.43      0.52       100
           3       0.43      0.40      0.42       100
           4       0.54      0.75      0.63       100
           5       0.53      0.64      0.58       100
           6       0.70      0.73      0.71       100
           7       0.65      0.70      0.68       100
           8       0.79      0.78      0.78       100
           9       0.78      0.64      0.70       100

    accuracy                           0.65      1000
   macro avg       0.66      0.65      0.65      1000
weighted avg       0.66      0.65      0.65      1000
```

The following table shows the performance of each comparison item on the f1-score.

Model	HOG feature	standarlize HOG feature	self-learning feature
SVM	0.41	0.60	-
KMeans	0.05	0.06	-
Gaussian Mixture Model	0.07	0.08	-
Neural Networks	0.56	0.48	-

Neural Networks + Transfer Learning	-	-	0.65

Among them, the `Neural Networks + Transfer Learning` model with the best f1-score has a higest f1-score(0.65).

4. Conclusion.

> Provide a summary for your method and the results. Also, provide your critical analysis; that is the shortcomings of your method and how they may be improved. [10%]

- summary: From the results in part 3, the effect of Neural Networks + Transfer Learning is the best, which can reach 0.65, but considering that the neural network requires higher training costs, SVM + standarlize HOG feature (f1 = 0.6) is used instead Xiang is also a good choice.

- critical analysis: 1. After using standarlize to process the HOG feature, the effect on the SVM model has been greatly improved. You can consider using the Principal Component Analysis / Linear Discriminative Analysis or other methods mentioned in the requirements to try more, in addition The hyperparameters of the SVM model can be further optimized, which may be able to achieve high results; 2. In the Neural Networks + Transfer Learning method, the selected ResNet18 model is pre-trained on the ImageNet image set, and the samples between the two data sets There are certain differences. You can try more pre-trained models to further take advantage of Transfer Learning and improve model performance.

5. References.

1. https://pytorch.org/docs/master/torchvision/models.html?highlight=resnet18#torchvision.models.resnet18
2. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmeans#sklearn.cluster.KMeans

4. https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html?highlight=gaussianmixture#sklearn.mixture.GaussianMixture
5. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html?highlight=mlpclassifier#sklearn.neural_network.MLPClassifier
6. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics