Assignment 2 - Group 15

Tutors: **Chen Chen, Zhiyi Wang, Henry Weld**

Group members:

Yichun Wang (UID: 470022177, unikey: ywan3103),

Jiachen Liu (UID: 500659847, unikey: jliu9852),

Shengwei Xiang(UID: 500682948, unikey: sxia9135)

Abstract

This report is about the research of image classification based on the CIFAR-100 dataset, by applying machine learning and data mining methods. In the rest of our report, we are going to introduce the problems intended to solve, evaluate the importance, discuss the methods used and the results we achieve. Also, a number of conclusions have been contained at the end of this report, with various improvement plans as well.

Introduction

In this report, we have chosen CIFAR-100 as our research dataset, which consists of 60,000 colour images with size 32 * 32, evenly split with 100 different classes. At the same time, these 100 classes are grouped into 20 superclasses. Our goal is to find the best classification method to correctly classify the images in the dataset as much as possible.

Problem intend to solve

In order to make sure the results we get are reasonable, it is necessary to find suitable data preprocessing methods and appropriate classification algorithms. On the other hand, when it comes to the long-time code running and considering the size of the dataset, we also have to reduce the images' dimension, which could be helpful for reducing running time.

After we have determined the choice of algorithms and methods, fine-tuning is also a very important step. We have to ensure the code could get favorable optimization, which guarantees it runs and classifies within a feasible time.

For the comparison among the chosen algorithms, similarly, we also need to consider many factors, such as accuracy, precision, recall and so on. The analysis and comparison of the results obtained by the algorithms is also a complicated process, which requires us to take it seriously.

Importance

Based on the problems we intend to solve in this report, after finishing these, we could have an excellent grasp of different advanced classifiers, be familiar with complex algorithms and methods, and obtain a chance of mastering external open source libraries. While through the study of the dataset itself, we can clearly realize the importance of choosing the correct classification algorithm for accuracy.

Previous Work

Big Transfer(BiT): General Visual Representation Learning proposed by Kolesnikov et al.(2020) is the literature that our team studied in this paper.

After implementing three different classifiers on the CIFAR-100 database, the literature's team conducted an in-depth investigation of the relevant literature on using databases. The literature proposed a new conception called upstream and downstream which corresponds to pre-training on a large dataset with supervised and fine-tuning target tasks' weight respectively. Below we summarize the main concept within the literature:

1) Large data scale and the combination of Group Normalization(GN) and Weight Standardization(WS) can contribute to the transfer learning in upstream pre-training.
2) A fine-tuning protocol for downstream tasks only demands one hyperparameter for each task. Resize the image into squares and choose to randomly flip the image horizontally without destroying the semantics of the label for pre-processing.

Through the literature, some experience in dealing with large datasets can be learned. Here we summarize the method that has not appeared in our report but is a better choice for future classification study:

1) The momentum stochastic gradient descent for massive dataset
2) Cut out the single object could save performance from the chaotic sense
3) Data enhancement scheme without destroying the label meaning: random horizontal image flip and zoom jitter
4) Utilize weight standardization in all convolutional layers.
5) Split and prominent the foreground objects that longest side corresponding to 80% of the width within the background.

Methods

Different techniques

- ***CNN (Convolutional Neural Networks)***

Convolutional Neural Networks is a kind of Feedforward Neural Networks that contains convolution and deep structures, which is one of the important representative algorithms of deep learning. CNN is also known as Shift-Invariant Artificial Neural Networks (SIANN) because of its representation learning ability, which enables CNN to carry out shift-invariant classification of input information according to its hierarchical structure.

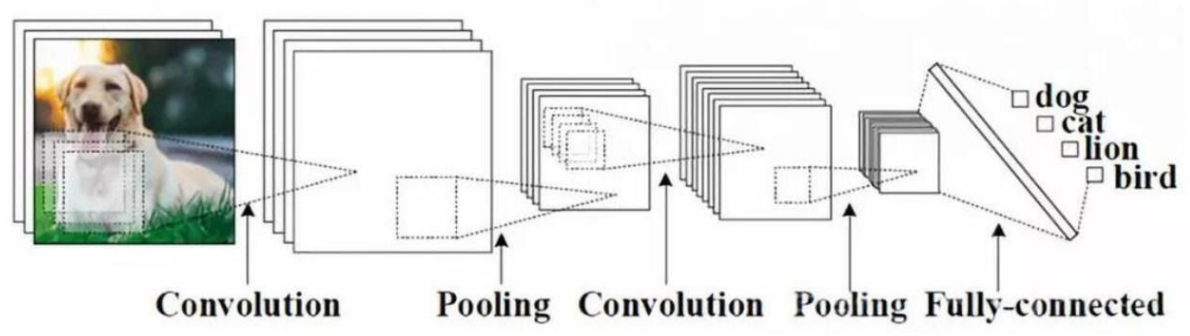


Figure: An overview of the basic convolutional neural network

Convolutional Neural Network has mainly composed of three parts: The first part is the input layer, followed by the hidden layer, which contains a large number of convolutional layers and pooling layers. The final part is the output layer, which consists of a fully connected multilayer sensor classifier.

From the structure of Convolutional Neural Networks, excluding the input layer and output layer, the core part of it is the hidden layer in the middle, including three common structures: convolutional layer, pooling layer and fully connected layer. In common constructions, the convolutional and pooling layers are unique to convolutional neural networks. Taking LeNet-5 as an example, the order is usually: input - convolutional layer - pooling layer - fully connected layer - output.

Convolutional Layer

The function of the convolutional layer is to extract features from the input data and it contains multiple convolution kernels. Each element of the convolution kernel corresponds to a weight coefficient and a bias vector, similar to the neuron of a feedforward neural network.

The parameters of the convolutional layer include the depth, stride and zero-padding. The three together determine the size of the output feature map of the convolutional layer, and they are the hyperparameter of the convolutional neural network. The size of the convolution kernel (the depth) can be specified as any value smaller than the size of the input image. The larger the convolution kernel, the more complex the input features that can be extracted.

Pooling Layer

After feature extraction in the convolutional layer, the output feature image will be passed to the pooling layer for feature selection and information filtering. The pooling layer contains a preset pooling function, whose function is to replace the result of a single point in the feature map with the feature map statistics of its neighboring regions. The pooling layer selects the pooling area in the same steps as the convolution kernel scanning feature map, which is controlled by the pooling size, stride and filling.

Pooling is a type of pooling model inspired by the hierarchical structure in the visual cortex. Its general representation is:

$$\boldsymbol{A}_k^l(i,j) = \left[\sum_{x=1}^{f}\sum_{y=1}^{f}\boldsymbol{A}_k^l(s_0 i + x, s_0 j + y)^p\right]^{\frac{1}{p}}$$

Fully Connected Layer

The fully connected layer in the convolutional neural network is equivalent to the hidden layer in the traditional feedforward neural network. The fully connected layer is located in the last part of the hidden layer of the convolutional neural network and

only transmits signals to other fully connected layers. The feature map loses the spatial topology in the fully connected layer, is expanded into a vector and passes the activation function.

- ***Random Forest***

Random forest is a standard supervised learning algorithm used for machine learning, which can be used for both classifiers and analytical regression. In the analysis of the cifar-100 data set, the algorithm was used to build a classifier model.

To introduce the principle of random forest, we must start with the decision tree, the fundamental component of the algorithm. The structure is generated by supervised learning from a set of data sets that have determined the label. Each internal node in the generated tree represents the judgment of an attribute, each branch represents the output of a judgment result, and each leaf node represents a classification result. The judgment at the branch and each node are determined by the features of the training set and its internal connections. The completed model can be realized after inputting a test sample, the sample will gradually extend from the root node of the tree to the leaf node according to the judgment, and then generate the judgment result according to the conclusion given by the leaf node. According to the above running process, it can be concluded that the key to the construction of the decision tree model is how to plan what features each node judges and what standard each node uses to make the judgment. There are currently three main judgment theories, ID3, C4.5, and CART.

The ID3 standard uses the principle of entropy to determine the choice of nodes and the judgments based on them.

$$Entropy=- sum [p(x_i) * log2(P(x_i)]$$

In the above formula, p(x_i) is the probability of feature I. The smaller the value of entropy (the minimum value is 0), the better the classification effect, and vice versa (the maximum value is one) the worse. The process of ID3 decision tree construction is to find the process of minimizing entropy. But the ID3 decision tree faces a serious overfitting problem because theoretically the data set can be continuously subdivided to the extreme so that the classification accuracy of the model on the training data is close to 100%.

To alleviate the overfitting problem in the ID3 standard, the C4.5 standard changed the information gain value in ID3 to the information gain rate, which means that blindly over-classification will not create the optimal solution for the model.

The difference between the CART decision tree and ID3 and C4.5 is that it selects the Gini coefficient to divide the features of the training set. Similarly, CART decision trees also face the problem of overfitting.

It can be seen that no matter what kind of decision tree is facing the problem of overfitting, to alleviate this problem, the model needs to be pruned to make it more

universal. As the name implies, pruning means to clean up branches that are judged to be redundant. Pruning methods are divided into pre-pruning and post-pruning. Pre-pruning means not building those branches that are identified as redundant during the construction of the decision tree, and post-pruning means after the decision tree is generated. According to the model's performance in the validation set, the redundant branch is processed. The random forest algorithm is an upgraded version of the decision tree. Its essence is to build a large number of imperfect decision trees and then determine the prediction result through the voting results of each decision tree. The random forest has better performance than a single decision tree, and the problem of overfitting can be avoided to the greatest extent by increasing the number of trees.

In the establishment of this random forest model, sklearn was used as a convenient tool by the group, and RandomizedSearchCV was used to find suitable parameters. This method provides a set of highly integrated interfaces so that users only need to limit the value range of the parameters to realize the adjustment parameters quickly, and cross-validation is also included in this method. However, because this method will perform repeated model verification on the parameter set on a large scale, it will cause a dimensional disaster and require a lot of computer resources.

- ***KNN (K-Nearest Neighbors)***

The third method has been chosen is the K-Nearest Neighbors(k-nn). Different from two previous algorithms, k-NN is a non-parametric classification under the supervised machine learning. A non-parametric algorithm can be understood as the model makes few assumptions about the data itself. In this case, it means the model structure established via k-NN is determined by given data. Such a feature determined the algorithm will have a few limitations itself. The main conception within the k-NN is that the algorithm assumes there is some adjacent proximity between each similar data. That means the similar data might have more likelihood to be clustered together.

To achieve the coding of k-NN, only 3 necessary things are required.

a) stored record sets
b) appropriate distance matrix to identify nearest neighbors
c) suitable value of K to achieve the filter the neighbors number

Reasoning

- ***CNN (Convolutional Neural Networks)***

As an image classification algorithm with decisive advantages, convolutional networks have the following advantages in image processing compared with general neural networks:

a) The input image and the topology of the network are in good agreement;
b) Feature extraction and pattern classification are carried out at the same time, and are generated during training at the same time;

c) Weight sharing can reduce the training parameters of the network, making the neural network structure simpler and more adaptable.

The reason for choosing convolutional neural networks is that the image classification algorithms before the advent of CNN, their performance is restricted by the extraction of features and the computational difficulties caused by a large number of parameters. However, the use of convolution, which simulates the working mode of the human visual system, has greatly reduced the number of training parameters of the neural network. This ensures that the corresponding features can be effectively learned from a large number of samples, avoiding the complex feature extraction process.

In this report, the CNN structure we used is designed based on Layers-13, including ten convolutional layers and a custom number of fully connected layers. The specific structure is as follows:

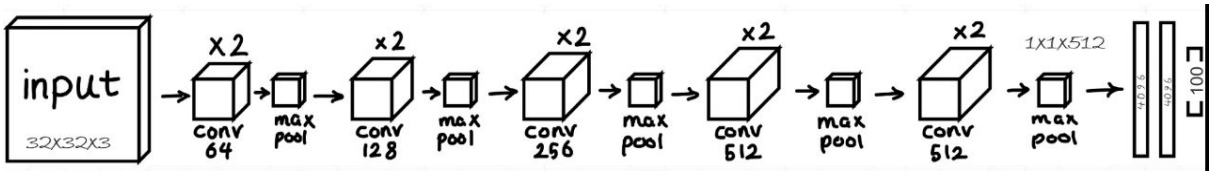


Figure: Layer-13 Convolutional Neural Networks

- ***Random Forest***

The random forest algorithm has many advantages. First of all, when the random forest is constructed, each tree inside will randomly select some features, so overfitting is largely alleviated, especially when the number of trees is quite large, the degree of overfitting of the random forest will be minimized. Second, the random forest algorithm has randomness when selecting features in the internal decision tree, so the classifier has good noise resistance. Besides, the random forest has a good classification performance for processing high-dimensional data sets, and it has good performance, whether it is dealing with discrete data or continuous data. (Nevertheless, I still do PCA preprocessing in the code, because this can increase the training speed)

Compared with classifiers other than neural networks, the accuracy of random forests is often more satisfactory. In terms of training speed, random forests are usually faster than neural networks.

- ***KNN (K-Nearest Neighbors)***

The third algorithm k-NN has it's unique algorithm concept different from the general probability algorithm. K-NN has a simple and mature algorithm theory.

Following are some advantages of k-NN:

a) High operability, k-NN is a relatively simple model in machine learning
b) Efficiency within multi-modal and large sample size classification problems

c) Fast on modal training
d) Not sensitive to outlier data

Chosen k-NN as the third algorithm with a theory of Occam's razor in mind, we hope to determine whether a simple theory can perform similar results compared to the complex method.

Pre-processing techniques

Normalization

For data preprocessing in convolutional neural networks, normalization is a necessary process. Normalization is to limit the data to a certain range after processing, usually within the interval [0, 1], so that the training data can have the same distribution. Due to the different distribution of the data, it will inevitably lead to different gradient drops in each dimension. It is difficult to iterate to the lowest point of the cost function using the same learning rate. After the normalization process, the gradient descent will become easier, while avoiding the gradient vanishing and exploding problems, and speeding up the convergence speed of the network.

Gray processing

All images in the CIFAR-100 dataset are color images (RGB images), which are highly recognizable, but if colors are included, the number of features and calculations will increase exponentially, which is not conducive to the speed of the algorithm. In order to identify objects, the most critical factor is the gradient. After grayscale, the dimension of the matrix decreases, the calculation speed is greatly increased, while the gradient information is still retained. Therefore, in the preprocessing part, grayscale processing is also an important part.

Contrast Limited Adaptive Histogram Equalization (CLAHE)

When there is a place in the image that is obviously brighter or darker than other areas, the ordinary gray image may not be able to identify the details of the place well. The application of the CLAHE algorithm can perform histogram equalization in a rectangular area around the currently processed pixel to achieve the effect of expanding local contrast and showing smooth area details.

One-hot Encoding

One-hot coding is the process of converting categorical variables into a form that is easy to use by machine learning algorithms. The method is to use N-bit status registers to encode N states, each state has its own independent register bit and at any time, only one of them is valid. In the classification problem, the data with existing categories is represented by X and the non-existent is represented by Y, where X is always 1, Y is always 0.

This method solves the problem that the classifier is not good at processing the classified data, and to a certain extent also plays a role in expanding the features.

PCA

PCA (principal components analysis) is a very common preprocessing method. The main purpose is to reduce the dimensionality of the data set without losing too much information. There are two main methods of PCA, one is to use eigenvalue decomposition, and the other is to use singular value decomposition. There are two main benefits of PCA, one is to improve the speed of training the model, and the other is to reduce the influence of noise value to a certain extent.

Experiments and Discussion

Description

Since the direct use of the test set provided by the original dataset will cause over-fitting, it is also conducive to directly finding the best parameter settings, which can be regarded as a kind of cheating. In order to avoid this situation, before data preprocessing, we split the original training set randomly by using k-Fold. The original training set is split into ten random parts, 90% of the data (45,000 data) is used as the new training set and the remaining 10% is regarded as a brand new test set. The new test set will be used to test the final results with the coarse labels and find the corresponding parameters of the algorithm.

After data importing, preprocessing, algorithm structure and final result prediction, the results of the three classification algorithms selected in this paper are as follows, including overall comparison:

- ***CNN (Convolutional Neural Networks)***

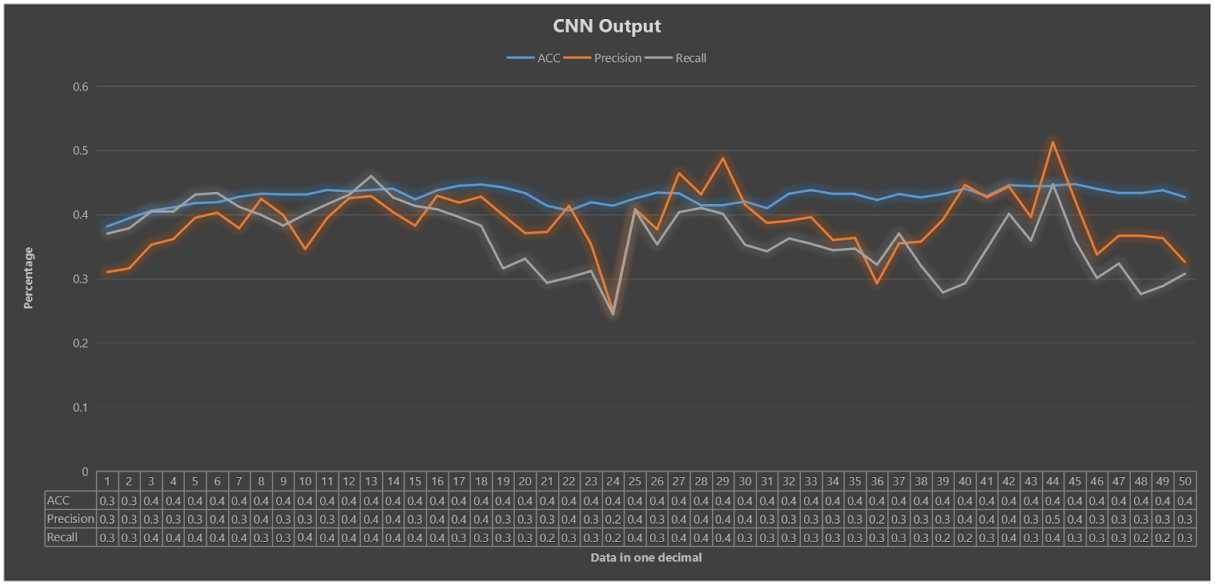
For Convolutional Neural Networks, all the performance evaluations run in this spec:

Hardware: Inter(R) Core(TM) i7-6700HQ CPU @ 2.60Ghz
RAM 12.0GB
Software: Python 3 Jupyter Notebook
Running time: Around 6 hours (390s for each epoch, 50 epochs in total)

In order to improve the speed and accuracy of the algorithm as much as possible, in the parameter setting of CNN, we set the number of convolutional layers and fully connected layers, including ten layers (2+2+2+2+2) and three fully connected layers. In the convolutional layer, we convolve the original 3-channel image into 256 channels step by step, and set the last layer of the fully connected layer to 50 categories. This improves the speed as much as possible while ensuring accuracy.

In the step of constructing the dataset object, we generate batch data for training the model, generate batch data in a loop and set the batch value to 225. The epoch value in the final training part is set as 50 as well, which is a value that is not too big or too small.

After 50 epochs, the independent result of each epoch is shown in the figure below, including each accuracy, precision and recall rate:



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
ACC	0.3	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Precision	0.3	0.3	0.3	0.3	0.3	0.4	0.3	0.4	0.3	0.3	0.3	0.4	0.4	0.4	0.3	0.4	0.4	0.4	0.3	0.3	0.3	0.4	0.3	0.2	0.4	0.3	0.4	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.3	0.4	0.4	0.4	0.3	0.5	0.4	0.3	0.3	0.3	0.3	0.3
Recall	0.3	0.3	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.2	0.4	0.3	0.4	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.3	0.4	0.3	0.4	0.3	0.3	0.3	0.2	0.2	0.3

Figure: CNN output with accuracy, precision and recall

From the above figure, we can clearly observe that when cross-validation is used, the classification prediction result (accuracy rate) for coarse_labels is almost maintained at a stable percentage, around 43%. In contrast, the results of precision and recall rate did not show similar stability in each independent epoch. But in general, there is not much difference from the results of accuracy rate and they are approximately maintained at the same level. The average values of precision and recall are 39% and 36%.

Depending on the results after 50 epochs, we decide to choose the 44th epoch as the best output, as this epoch has the highest precision and recall percentage. In order to facilitate comparison with other classification algorithms, the result of confusion matrix generation is as follows:

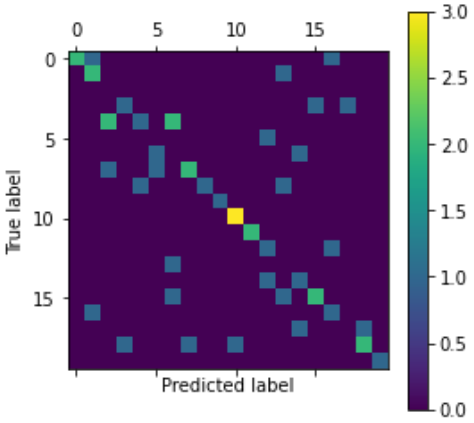


Figure: CNN confusion matrix

- ***Random Forest***

The construction of the random forest model runs on a machine with the following configuration:

Hardware: Intel(R)_Core(TM)_i7-10750H_CPU_@_2.60GHz
RAM 16.0GB
Software: Python 3 Jupyter Notebook
Running time: It takes about one hour to find the parameters, and it takes about five minutes to draw a conclusion from the model analysis test set.

By setting the parameters of RandomizedSearchCV to the following interval, an optimal parameter combination is finally obtained as shown in the following table.

Parameters selected in random forest:

n_estimators	600, 700, 800, 900, 1000
max_features	auto, sqrt
max_depth	6 to 20
min_samples_split	2, 5, 10
min_samples_leaf	1, 2, 4
bootstrap	True, False

Parameters selected in RandomizedSearchCV:

n_iter	20
cv	10

After training, the result including classification report and confusion matrix is shown as following graphs.

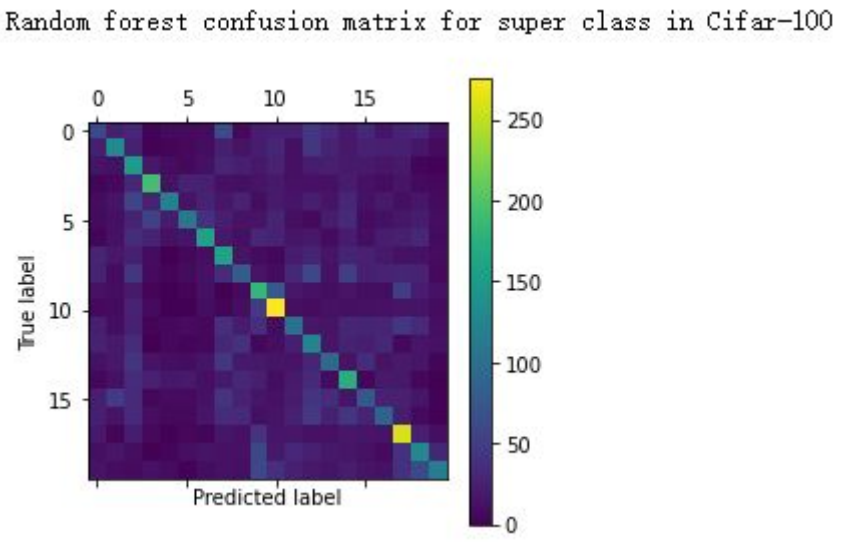


Figure: Random forest confusion matrix

Random forest classification report for super class in Cifar-100

	precision	recall	f1-score	support
0	0.17	0.12	0.14	502
1	0.32	0.27	0.29	501
2	0.21	0.33	0.26	474
3	0.44	0.41	0.42	481
4	0.44	0.24	0.31	504
5	0.38	0.22	0.28	497
6	0.37	0.30	0.33	524
7	0.25	0.34	0.29	468
8	0.19	0.15	0.17	523
9	0.28	0.37	0.31	500
10	0.40	0.54	0.46	510
11	0.22	0.21	0.21	505
12	0.20	0.27	0.23	475
13	0.21	0.20	0.21	472
14	0.27	0.34	0.30	504
15	0.17	0.15	0.16	503
16	0.20	0.18	0.19	502
17	0.36	0.48	0.41	544
18	0.25	0.27	0.26	494
19	0.41	0.23	0.29	517
accuracy			0.28	10000
macro avg	0.29	0.28	0.28	10000
weighted avg	0.29	0.28	0.28	10000

Figure: Random forest output with precision and recall

- ***KNN (K-Nearest Neighbors)***

For K-Nearest Neighbors, following is the code running environment detail:
Hardware: Inter(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
RAM 16.0GB
Software: Python 3 Jupyter Notebook
Running time: To find the best k in the range of 6 to 14 under the 10-fold, it takes almost a night. But there are only 7 minutes to have the final solution.

Parameter setting: By implementing the k-fold in 10, the best k value has been found in a range of 6 to 1. The figure below is the process of the appropriate k being found.

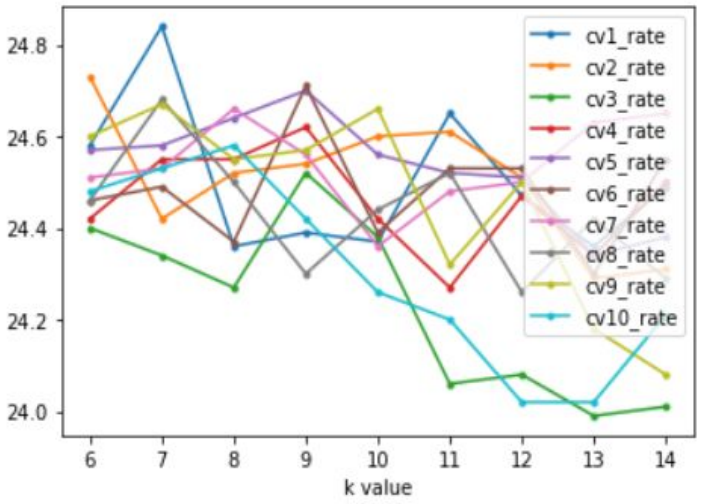


Figure: k-NN output with accuracy

For the confusion matrix, precision and recall of k-NN, please refer to the following two figures.

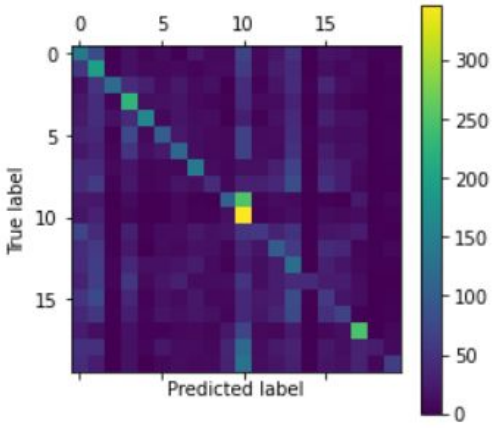


Figure: k-NN confusion matrix

	precision	recall	f1-score	support
0	0.17	0.28	0.21	500
1	0.18	0.39	0.25	500
2	0.64	0.24	0.35	500
3	0.29	0.46	0.35	500
4	0.54	0.32	0.40	500
5	0.32	0.21	0.25	500
6	0.30	0.23	0.26	500
7	0.35	0.29	0.32	500
8	0.20	0.09	0.12	500
9	0.34	0.22	0.26	500
10	0.22	0.69	0.33	500
11	0.20	0.11	0.14	500
12	0.20	0.20	0.20	500
13	0.12	0.25	0.17	500
14	0.65	0.09	0.15	500
15	0.11	0.12	0.11	500
16	0.18	0.14	0.15	500
17	0.48	0.50	0.49	500
18	0.54	0.05	0.09	500
19	0.48	0.13	0.20	500
accuracy			0.25	10000
macro avg	0.32	0.25	0.24	10000
weighted avg	0.32	0.25	0.24	10000

Figure: k-NN output with, precision and recall

- ***Comparison***

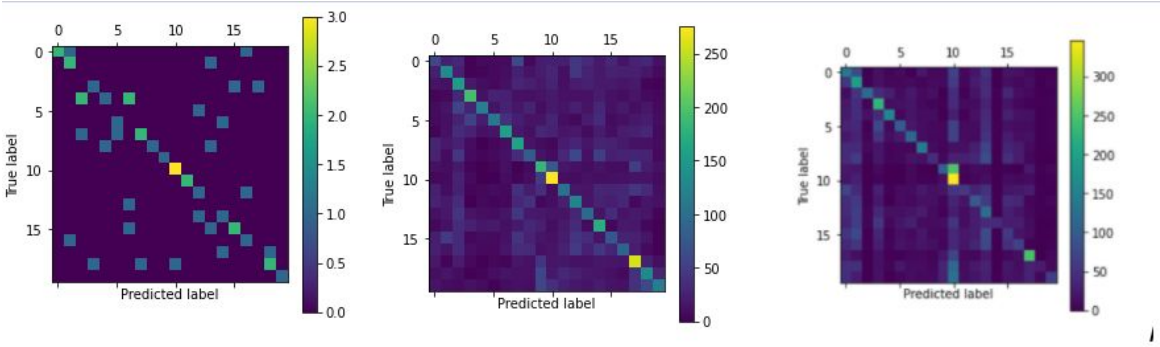
	Accuracy	Precision	Recall
CNN	**0.4279**	**0.3887**	**0.3642**
Random Forest	**0.28**	**0.29**	**0.28**
KNN	**0.2504**	**0.32**	**0.25**

Figure: Table of the results of three classification algorithms

After running the three different classification algorithms mentioned above, we conducted a specific comparative analysis of the different accuracy, precision and recall obtained. The results of the comparison are shown in the table above.

Obviously, CNN performs better than the other two algorithms, whether it is accuracy, or precision and recall. The performance gap between the random forest and the KNN model is very small. KNN is slightly inferior to the random forest, but it is better than the random forest in the precision part.

In order to further study the comparison, we also chose to compare the confusion matrix of the three, which could observe the difference between them more clearly:



We have observed that in the results of the confusion matrices, the performance of CNN is not as good as expected. On the contrary, the matrix images of random forest and KNN seem to be more accurate. This may be due to problems in the CNN matrix export process. The amount of data is not unified with the other two algorithms. So the comparison of the confusion matrices is difficult to get a reasonable conclusion.

Overall, compared with the other two algorithms, CNN performs better on the image classification problem based on the CIFAR-100 data set. Therefore CNN is going to be our preferred classification algorithm.

Discussion

Although from the results of algorithm comparison, CNN is undoubtedly the preferred classification algorithm, its biggest flaw is that the overall running time is too long. The reason why CNN needs such a long time to perform calculations is not only because of the need for multiple cycles of training, but also because of the small size of the picture and the characteristics of multiple classifications, which makes the project difficult to test.

However, considering that CNN can extract local features and increase the amount of computational data, the effect on image classification is significantly better than other algorithms. This is also the main reason why we finally chose CNN as the final algorithm.

In terms of evaluation results, the evaluation indicators of the random forest model are better than the knn model, but not as good as the CNN. CNN is more suitable as a prediction model for the cifar-100 data set. However, compared to the CNN model, the advantage of the random forest model is that the model is built much faster than CNN, even if the random forest uses RandomizedSearchCV. And after the model is established, the random forest can quickly predict the test set, while CNN takes a long time. In short, the random forest method is inferior to CNN in all indicators of prediction results, but it has higher efficiency.

At last, the algorithm k-NN is not recommended by this paper compared to two others. Although it is known as the easiest machine learning algorithm, the inherent laziness from the algorithm concept itself has serious limitations on insufficient and unbalanced distributed data sets. In terms of time, knn is considered to be the most

time-consuming algorithm from the other two methods. The huge calculation process destined from the algorithm itself is obviously slower than CNN and random forest. And with the joint of k-fold, the consuming time itself will at least multiplied by k times. Such limitations are inherent. Thus, this paper would either recommend CNN with higher-accuracy or random forest with reasonable time consumption. And clearly, from the figure above, the overall performance of k-NN in terms of time, accuracy and even recall is disappointed.

Reflection

For the research of CNN classification algorithms, we can clearly feel the impact of parameter settings on the results, and there is great room for improvement and optimization in the architecture of the algorithm. A huge amount of calculation is not suitable without sufficient excellent hardware support, which means that the team should optimize the data preprocessing process as much as possible in a limited operating environment so that it can better fit the training model to improve the running speed and accuracy.

For the random forest model, our team needs to prepare a more suitable preprocessing method to make the model fit better. In addition, in the adjustment of parameters, we need to make a better estimate of the range of various parameters to optimize the speed of RandomizedSearchCV.

In terms of k-NN, the team thinks that there might be an improvement when changing the distance formula to match the current datasets. On the other hand, for large calculation problems, the K-dimensional tree could be a choice for reducing the operation time.

Conclusion and Feature Work

Conclusion

In this assessment, our group trained three different classifiers for this subject research. The choices cover unsupervised models, supervised models, and deep learning models. This gave our team a deep understanding of the learning of classifiers in machine learning. In addition, we have accumulated a lot of experience in data preprocessing, parameter adjustment and conclusion evaluation methods, which will be of great benefit to our future research on machine learning.

Feature work

Compared with some mature models that predict cifar-100, data preprocessing is our biggest shortcoming. Therefore, in order to predict more accurately on this data set, in the future we will try to study more pictures and processing aspects. the study. Including but not limited to picture flipping, c algorithm etc. In addition to the in-depth research on the cifar-100 data set, we will continue to understand some regression

prediction-related algorithms, broaden our knowledge in the field of machine learning, and lay the foundation for future learning of deep learning algorithms.

References

Alex. K, Vinod. N, and Geoffrey. H, 2009, CIFAR-100 dataset, available at <https://www.cs.toronto.edu/~kriz/cifar.html>

Alexander, K; Lucas, B; Xiaohua, Z; Joan, P; Jessica, Y; Sylvain, G and Neil, H, 5th of May 2020. "Big Transfer (BiT): General Visual Representation Learning", available at: <https://arxiv.org/pdf/1912.11370.pdf>

Welch, N 2000, "Toward an understanding of the determinants of rural health", viewed 9 January 2002

Appendix

Steps for running code:

CNN:

1. In order to run the code with the final test dataset, when you run the code, please replace the validations. In details:

```
### Import dataset
def unpickle(file):
    import pickle
    with open(file, 'rb') as p:
        dict = pickle.load(p, encoding='bytes')
    return dict

meta = unpickle("C:/Users/Administrator/Desktop/5318 Group/cifar-100-python/meta")
train = unpickle("C:/Users/Administrator/Desktop/5318 Group/cifar-100-python/train")
#test = unpickle("C:/Users/Administrator/Desktop/5318 Group/cifar-100-python/test")
```

   Please remove the '#' before the 'test' and rerun this cell.
2. Remove the split code and rerun the cell so it can ensure you are running the original test dataset (As we are using validation for finding parameters):

```
### Split training set
from sklearn.model_selection import train_test_split

train_x = train[b'data']
train_y = train[b'coarse_labels'] # Use coarse_labels for classification
#test_x = test[b'data']
#test_y = test[b'coarse_labels']

train_xs, test_xs= train_test_split(train_x, test_size=0.1, random_state=0, shuffle=True) # 10-kFold
train_ys, test_ys= train_test_split(train_y, test_size=0.1, random_state=0, shuffle=True)
```

```
### Transfer images into RGB
train_x=train_x.reshape(-1,3,32,32)
train_x=np.rollaxis(train_x, 1, 4)
train_xs=train_xs.reshape(-1,3,32,32)
train_xs=np.rollaxis(train_xs, 1, 4)
test_xs=test_xs.reshape(-1,3,32,32)
test_xs=np.rollaxis(test_xs, 1, 4)
#test_x=test_x.reshape(-1,3,32,32)
#test_x=np.rollaxis(test_x, 1, 4)
```

Remove '#' before the test_x and test_y, and add '#' before the 10-kFold code, then rerun the cell. Remove '#' before the test_x marked by red arrows, then rerun the cell.

3. Replace the old dataset with new ones:

```
### Build the dataset object
train_db = tf.data.Dataset.from_tensor_slices((train_xs, train_ys))
train_db = train_db.map(preprocess).batch(225) ## Pack bulk data into a batch
test_db = tf.data.Dataset.from_tensor_slices((test_xs, test_ys))
test_db = test_db.map(preprocess).batch(225)
```

Replace the train_xs and train_ys with train_x and train_y, and replace test_xs and test_ys with test_x and test_y as well.

4. Now you can run the whole code and the result should contain the original training set and test set instead of the validations.

Random forest & KNN:

1. Before running the code, make sure that the cifar-100-python folder, including train, mate and test files, is on the same path.
2. For the random forest part:
   To experience the process of finding parameters, run all the codes in the random forest part except the last code cell. In order to simply use the test set to run the random forest model, run the last code cell of the random forest part alone
3. For knn part:
   Simply run all the cells in the KNN section in sequence. If you don't want to waste too much time on K-fold, you can skip the K-fold cell. Because the selection result of the best parameter is already written in the next cell.