COMP5318 - Machine Learning
Assignment 2 Report

Ye, Yuan yyua5381@uni.sydney.edu.au - 460442862
Handuo, Mei hmei6896@uni.sydney.edu.au - 470471467
Ruoyu, Zhuang rzhu9225@uni.sydney.edu.au - 470132373

June 4, 2018

1 Abstract

There exists many methods to handle with the classification problems with machine learning and data mining. In this paper, three different classifier methods: K-Nearest Neighbors (KNN); Random Forest (RF); and Convolutional Neural Network (CNN) are introduced and implemented on solving multiple classification of the real images dataset: CIFAR-10 dataset. In the pre-processing phase, we apply Principal Component Analysis (PCA) to perform feature extraction for KNN and RF, in order to improve the efficiency and accuracy of these two classifiers. For the CNN, the convolutional filters used in each of layers has similar purpose as the PCA, but it project the data onto a higher dimensional space rather than lower, to abstract the higher level features of images. After that, we evaluate and compare the performance of the models according to the specific metrics including accuracy, confusion matrix, cost-sensitive measure and computation runtime. As the result of the evaluation, CNN outperforms the other models with much higher accuracy of more than 90% within 100 epochs.

2 Introduction

2.1 Problem Description

Along with the fast development and progress of the science technology around the world in recent years, a mass of new data such like text, images and videos are generated in human life all the time with the unstoppable exponential growth rate. Classification is the instinct and demand of human behavior, the unordered and unlabeled images data information will obtain the meaning and value after classified by people. Image classification, a classic and essential problem in computer visions, thus becomes a hot topic with widely applications such as tagging and labeling in social Apps, auto-driving techniques, remote sensing and GIS. [20]

As a classic machine learning problem, most of the well-known classification algorithms such as k Nearest Neighbors and Random Forest can be applied to this problem. However, Convolutional Neural Networks is one of the most popular methods which has been proved efficient and precise in many cases. Hence it is a valuable practice to implement and tune them on the problem, and it is also of great significance to compare and analyze the performances of various classification algorithms as well as the mechanisms supporting the outcomes.

2.2 Dataset Exploration

In this assignment, we choose the CIFAR-10 dataset to solve the images classification problem, which can be found from the URL: https://www.cs.toronto.edu/~kriz/cifar.html. CIFAR-10 dataset is a labeled subset of 80 million tiny images dataset, which contains 60000 32x32 color images in 10 classes, with 6000 images per class. There exists 50000 training images and 10000 test images[14]. The images are the real pictures of objects, which belong to 10 classes - airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The classes are regarded as completely mutually exclusive. The Figure 1 on page 2 shows 10 images in

each class from CIFAR-10[15]:

Figure 1: CIFAR-10 Sample Images

This data is one of the most widely used datasets for machine learning research[6], as recognizing objects in photos or videos is the typical images classification problem. The similar datasets of CIFAR-10 involve CIFAR-100, ImageNet, SVHN, etc.

3 Previous Work

There exists many approaches handling with the images classification problems. Different influences of factor make the classification a complicated procedure.

D.Lu and Q.Weng suggest that designing the suitable pre-processing procedure of raw dataset is the prerequisite for a successful images classification outputs[21]. They support the principal component analysis approach to extract only the features that are relative useful in order to reduce the data redundancy.

The original K-Nearest Neighbors rule is a non-parametric method for pattern classification which was first introduced by Fix and Hodges in 1951[8]. Hao Zhang, A.C. Berg, and M. Maire achieved the classification accuracy of 59.05% at 15 training images per class, and 66.23% at 30 training images while applying K-Nearest Neighbor classifier with limited sampling on object recognition of images dataset Caltech-101[22]. D.S.Guru, Y.H.Sharath and S.Manjunath implement and obtain the meaningful experiment and performance with effective computation runtime, by applying K-Nearest Neighbors in classification of flower images with textual features in color level co-occurrence matrix[12], which has the similar form to the dimensions of CIFAR-10 dataset.

Random forests is an ensemble learning method for classification, it implement by constructing a multitude of decision trees in training and outputting the class of the individual trees. The algorithm was first proposed by Ho in 1995[13]. It is supported by many experimental results to perform well in high-dimension classification[2]. Recent research are concentrated on further improvements of ensemble optimization. The error bound witness dramatical decrease by applying the feature weighting methods proposed by B. Xu et al in subspace sampling and tree selection[26] for image classification, which is proved to be able to reduce the computational expenses as well.

In the past few years, various version of Convolutional Neural Networks have outperformed any other classifiers on the CIFAR 10 data set. Zeiler et al achieved 84.87% at the validation set by using stochastic pooling technique[27], which randomly picking the activities within the pooling region rather than deterministic pooling operations. In 2014, Ian et al developed a new activation operation namely Maxout Activation[10] and replaced the widely used rectified linear unit activations. They obtained accuracy of 88.32% without data augmentation, and 90.62% with data augmentation. So far, the state of art classifier on the CIFAR 10 dataset is a Convolutional Neural Network which involved a technique called fractional max-pooling[11], and this model achieved 96.53% accuracy on 100 tests.

For the evaluation phase, confusion matrix is a important concept from machine learning. It contains information about actual and predicted classifications done by a classification system[5]. It provide practicer and researcher the directly visualization to observe the model performance in different classes of the methods.

4 Methods

In this study, we implement and evaluate three classifiers to deal with the image-based multiple classification. First we try to apply K-Nearest Neighbors and Random Forest with relatively simple hypothesis. Then we build the Convolutional Neural Network model, the most commonly application to explore visual images. After that, we fine tune different hyper-parameters within different models with outputs and reasonable explanations for hyper-parameter selections are provided. Addition-

ally, the number of components in the feature engineering method Principal Component Analysis is also treated as a hyper-parameter to be tuned for the classifier. The comparison among different methods is realized with evaluation indexes such as precision, recall, f1-score and confusion matrix, which reserves the variance among labels as well.

4.1 Pre-processing

4.1.1 PCA

Principal components analysis is a widely utilized method for dimension reduction. The idea is to apply SVD decomposition on the centered data and choose first n principal components with largest singular values. It actually performs linear transformation of original features, and can be regarded as a low-dimensional projection of feature space. Based on our study, PCA are able to remove noise from images[1].

- Advantages: Considerably reduce the dimensionality while lose little variance.
- Disadvantages: Very computationally expensive in both time and memory due to SVD decomposition.

4.1.2 Data Scaling

Each color image is constructed by three color, which are red, green, and blue correspondingly, and each color must fall in the value range from 0 to 255. We scaled the train and test images by simply divided by 255, thus all images now have the pixel values from 0 to 1.

4.1.3 One Hot Encoding

The original train and test Y was a one dimensional array with the length of 50000 and 10000 correspondingly, where each value within these arrays indicates the corresponding class to the instances. Since the CNN used softmax activation and cross-entropy as the loss function, it required the input $\hat{Y}$ and Y are one hot vectors in order to compute the cross-entropy loss across all training examples. One hot vector encoding representation used a vector to represents a class. The vector made by 1 and 0, where the 1 indicates the true class which it belongs to. For example, an integer 6 will be modified into the following format if using one hot vector:

$$[0, 0, 0, 0, 0, 0, 1]$$

The location of 1 has the index of 6, which now represents its class.

4.1.4 Data Augmentation

In order to obtain a robust model that can generalized well on both train and test data set, we have implement several data augmentation techniques, which including randomly zooming, shifting, rotating and flipping. It is very natural to think that, for most of object in the image such as a dog or a cat, human can almost always identify this object even if the image been rotated, stretched, or horizontally and vertically shifted. The same idea have been proved to be useful for the machines. According to several past experiments [10][11][25], such data augmentation can largely boosted the performance of the model by around 3%.

4.2 Classification Techniques

4.2.1 K-Nearest Neighbors

K-Nearest Neighbors Classifier is the one of the most simple and basic non-parametric classification method in Machine Learning and Data Mining, which consists of the geometric distance between k closest training examples in the feature space[24], which is robust to treat the noisy and unbalance dataset. The CIFAR-10 dataset contains enough sample size and reasonable samples distribution per class so that KNN suppose to work fine on object recognition for our dataset.
It requires three elements: The set of labeled samples, distance metric to compute distance between samples, and the number of k nearest neighbors to retrieve. Given the images dataset, sends a query image sample to the algorithm server, while the server returns k neighbors of query image sample from within dataset where each distance between the k neighbors and query image sample is minimal in dataset, that is, the k neighbors are the k nearest neighbors to query image[3]. The

algorithm approach collect the k neighbors samples $(X_1, X_2, ..., X_k)$ of query image sample (Y) and classify the Y according to the closest distance within $X_1, X_2, ..., X_k$:

$$C(Y) = MinDistance_{i=1}^{k}(X_i, Y)$$

There exists many of distance computation methods between query image sample and k neighbors samples. Suppose Y is the query image sample with p number of features $(y_1, y_2, ..., y_p)$, and X is the one of k neighbors samples with p number of features $(x_1, x_2, ..., x_p)$. One of the most common methods is Euclidean Distance which the formula is define as[24]:

$$Euclidean\_D(Y, X) = \sqrt{\sum_{i=1}^{p}(y_i - x_i)^2}$$

4.2.2 Random Forest

Random Forest is actually an ensemble method, which build a model comprised of decision trees[13]. Its base model, decision tree, generates a series question about features, which presented as intermediate nodes in the tree[19]. Following the questions along the nodes, samples can be finally classified into labeled leafs. The decision tree is built up by recursively partitioning the dataset into subsets with as pure labels as possible. Let s denote split schema, x_j and t_k represent certain feature and its corresponding threshold on node k, respectively. The partition could be described as

$$T_{left}(s) = (x, y) \in T : x_j \leq t_k$$

$$T_{right}(s) = (x, y) \in T : x_j > t_k$$

The criteria for partitioning can be maximum depth, minimum instance per node, or the purity of the tree. And there are several possible choices for purity calculation, one of which is Gini index:

$$H(T) = 2\prod_{i=0}^{1}\frac{|\{(x, y) \in T : y = i\}|}{|T|}$$

Random Forest first sample the data with bootstrap method as the input for each tree and the trees are established with the aforementioned rule. Then the trees are put into an ensemble with voting as weights, which is the final model.

Since Random Forest contains a set of decision tree produced by different sampling subset of data, it is able to avoid the possible low bias in single decision tree, which might eventually leads to over-fitting. Hence in most case, Random Forest can outperform the model of single decision tree, also true for our dataset, where RF can obtain an accuracy nearly twice than single decision tree.
The time complexity of RF mainly depends on the size of input as well as trees. So we consider number of dimensionality of input data d and number of trees n as the hyper-parameters to be tuned further.

4.2.3 Convolutional Neural Network

Convolutional Neural Networks are the Neural Networks with at least one, at most all convolutional layers, and this class of Neural Networks is particularly useful for image recognitions. The various version of CNNs now have domain the Image Net Large Scale Visual Recognition Challenge (ILSVRC) since 2012[16].
Convolutional Neural Networks are very similar to the Neural Networks; they all constructed by multiple layers with bunch of neurons in each layer. The neurons are actually the weights, and generally they are tiny random numbers at the beginning. The network then performs the forward propagation and back propagation to train these weights, intend to fit the latent non-linear function. The forward propagation performs the dot product of the inputs and weights, then typically follows with some activation functions. The back propagation updates the weights in each layers by taking their derivatives respect to the defined loss function, and the magnitude of each update are controlled by the learning rate α. The general form of the linear and non-linear forward process is defined as:

$$Z^{[l]} = {W^{[l]}}^T A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(Z^{[l]})$$

The l indicates the corresponding layer, for example, $Z^{[l]}$ is the linear output in layer l, and the $g(.)^{[l]}$ is the activation function at the same layer which performs non-linear transformations to the linear output. Training a neural network equivalents to finding the proper weights W and bias b in each layer. When we say proper, it

means the local optimal solutions for the weights and bias, not the global optimal. Optimizing the weights and bias in the neural network generally use the gradient based learning methods, specifically, gradient descent methods, which iteratively updates the weights and bias proportional to their gradients.
A Neural Network generally take a vector or vectors as input, while the CNNs take binary (black and white) or colored images as input and treat each pixel as a feature. An image has two to three dimensions depends on its color, each corresponding to the width, height, and color channels. Images stored as matrix or tensors in the computer, where each number in the matrix or tensor represents a pixel value of the image, falls in the range of 0 to 255. A typical CNN generally formed by convolutional layers, pooling layers, then follows by one or two fully connected layers[7], and finally output the regression or probabilistic results. More properties of CNNs and the difference to the ordinary Neural Networks will be talked in the next several sections.

Convolution Operation

The name of Convolutional Neural Network is because they used multiple convolutional filters to conduct the convolution operations to the inputs to each convolutional layer. Suppose a 3×3 convolutional filter, it performs the dot product to a same size area on the inputs and get a scalar output, and then do the same operation step by step goes from left to right, top to bottom according to the stride size s. Simply speaking, each filter can be thinks as containing some particular features such as a color dot or a shape. By adding more convolutional filters, the model can have the ability to abstract more higher-level features, and later reconstruct these features to make the prediction[9].

Small Region Connected

While the ordinary Neural Networks have all neurons in each layers fully connected to the inputs and outputs to that layer, each neuron in a convolutional layer generally only looks at a small regional area in the previous layer. Such property is one of the major advantage of why CNNs can perform well on the image recognition task, as it took the spatial relations into consideration.

Pooling

Put a pooling layer immediately after the convolutional layers is another widely used technique used in many CNNs[17]. Generally, pooling conduct down sampling operations, resulted in the reduction of dimension of the tensors during the computation progress, as well as the total trainable parameters in the model. However, some recent research [25] conducted the experiment to test the necessity of the pooling layers in the CNNs. Surprisingly, the test classification errors dropped after replaced the pooling layers by a another convolutional layer with increased stride size. The author also stated that it will loss exactly 75% of activations from the input if using pooling operations. Therefor, we used similar operations in this report to conduct the experiments on CNN for classification task on CIFAR 10.

4.3 Design Choices

- All coding work are run in Python programming language, some major libraries are listed: numpy, matplotlib.pyplot, sklearn, keras, etc.
- Uploading the dataset by importing keras.datasets.cifar10 library. It is exactly equal to our target dataset downloaded from the URL.
- Separately employing the K-Nearest Neighbors and Random Forest approaches with original dataset (full 3072 features) and processed dataset by applying Principal Components Analysis(reduced features). Comparing the effect of performance and speeding of run time in training dataset on each combinations of methods and datasets. No need to use PCA for dataset in Convolutional Neural Network.
- Fine tuning the hyper-parameters of models in each classifiers, the three models with optimal performance and run time should be adopted

in the three classifiers. Implement the experiments with 10-fold cross validation exercise in training dataset.

- Unlike any other machine learning models which generally have less than 10 hyper parameters to select, tunning hyper parameters can be tough for a CNN since there almost no limitation on the total number of hyper parameters and parameters. Therefore, we started our CNN based on some successful CNN architectures in the past. The architecture of our CNN was inspired by Springenberg et al[25] and Lin et al[18]. We stacked several convolutional layers together, use the stride size of 2 for 2 convolutional layers to conduct the dimension reduction instead of using max pooling, and used 1×1 layers at the very end to reduce the depth of the output before softmax activation operation. Moreover, we have used the leaky rectifier activations for all layers instead of rectifier linearity activations. In order to generalize the performance of our model on both train and test set, we added 50% dropout to the convolutional layers immediately after dimension reduction.

- KNN and RF are fine to run on the environment of our local laptop. Deploying the Ubuntu Linux environment on the Google Cloud and install the necessary Python and Nvidia Cuda packages, in order to boost the speed of training the Convolutional Neural Network. We used a Nvidia Tesla P100 GPU to train CNN.

- As RF can be easily parallelized, we set up some experiments on Spark via lab cluster to compare its runtime with single laptop.

- Evaluating the three different classifiers in test dataset and compare the performance included accuracy, precision, recall confusion matrix, and cost-sensitive measure. Choosing most the appropriate model.

5 Experiments & Discussion

5.1 Experiments Results

5.1.1 Results of KNN

In the first step, the original dataset are transform to the several datasets in different shapes of features by PCA function with the different values of component. We practice the KNN models with default constant hyper parameters (neighbors k=5, weight=uniform) on the original and transformed datasets, which the algorithm is imported from scikit-learn library in python, the results are listed below(10-fold CV output mean values):

PCA	**Mean Acc**	**Mean Runtime**
15 comps	36.57%	19s
20 comps	39.21%	26s
30 comps	40.19%	38s
35 comps	40.17%	41s
40 comps	**40.38%**	**43s**
45 comps	39.44%	53s
50 comps	39.34%	54s
100 comps	38.09%	1m40s
200 comps	36.39%	3m01s
all feature	33.84%	45m01s

Table 1: KNN in diff. PCA comp. in 10-fold CV

Table 1 on page 6 show that the KNN model output the highest mean accuracy 40.38% and appropriate runtime 43 seconds in 10-fold cross validation exercise on datasets with 40 PCA components. The more components, the more runtime cost. It is the fact that PCA are able to improve the accuracy of the KNN model as it reduces overfitting[1]. Therefore, PCA will be applied in KNN.
In the next step, based on our above experiments, we will practice the KNN model in dataset with 40 PCA components by fine tuning the different hyper-parameters to optimize the best performance. The key parameter is the number of neighbors k. Another parameter is the weight function used in prediction it has two strings values[23]:

- uniform: Uniform weights. All points in each neighborhood are weighted equally.

- distance: Weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

We output the results in combination of parameters as follow(10-fold CV output mean values):

uniform		
Neighbors k	**MeanAcc**	**MeanRuntime**
10	41.75%	30s
15	41.90%	27s
16	42.04%	32s
17	42.17%	30s
20	42.06%	32s
distance		
Neighbors k	**MeanAcc**	**MeanRuntime**
10	42.59%	28s
15	**42.77%**	**30s**
16	42.73%	29s
17	42.56%	29s
20	42.30%	32s

Table 2: KNN in diff. Param. in 10-fold CV

Table 4 on page 8 show that the KNN model output the highest mean accuracy 42.77% and appropriate runtime 30 seconds in 10-fold cross validation exercise on datasets with 15 neighbors and distance weight function. The values of neighbors k have relative influence on KNN models and gives the highest level in specific value. The weight function in different approaches output the similar runtime but different performance. Distance weight function can work better on KNN.
As the result, we decide to choose the model with hyper-parameters (15 neighbors, distance weight function) in training with 40 PCA components dataset as the optimal K-Nearest Neighbors classifier.
We execute the confusion matrix plan for optimal K-Nearest Neighbors classifier (15 neighbors, distance weight function, 40 PCA components) as follows:
The Figure 2 on page 7 represent the Confusion Matrix of KNN on test dataset. We observe that the classes airplane, frog, and ship have acceptable

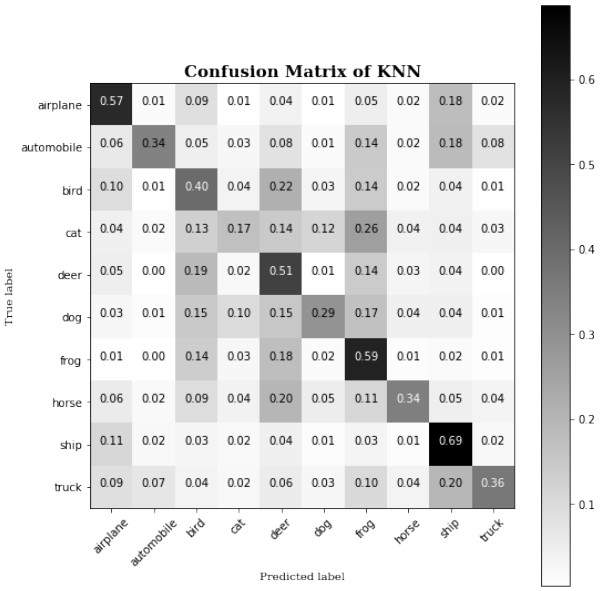

Figure 2: Confusion Matrix of KNN

accuracy, the numbers of correctly classified are exceed half of subsamples in these class. The classes automobile, bird, cat, deer, dog, horse, and truck have the accuracy below the average. Some classes have higher chance rather randomly been misclassified to the particular class. The true classes as cat, and dog have very low accuracy and they are likely to be misclassified to under predicted classes as others animals.
The cost-sensitive measure plan for optimal K-Nearest Neighbors classifier (15 neighbors, distance weight function, 40 PCA components) are showed as follows:

KNN	precision	recall	f1-score
Average	0.47	0.43	0.42

Table 3: Cost-Sensitive Measure of KNN

A more detailed performance of KNN on test dataset is pictured on page 9 (Table 6) . The average precision, recall and f1-score are all below the 50%.

5.1.2 Results of RF

For Random Forest, we focus on two hyper-parameters, i.e. the number of reduced dimensionality d from PCA and the number of trees n in the ensemble. We split the train data into test and validation set for parameter tuning, and 10-folds cross validation grid search of these two parameters are used in tuning. Table below is several combinations picked from all results (runtime here denotes the average runtime):

d	**n**	**Accuracy**	**Runtime**
50	10	37.22%	30s
50	100	41.49%	52s
100	100	45.65%	1min12s
100	500	49.23%	4min44s
100	1000	49.66%	8min40s
150	1000	49.28%	11min28s

Table 4: RF in diff. Param

It can be found that with 100 reduced features and 500 to 1000 trees, the accuracy fluctuate slightly around 49%. Considering the runtime, $d = 100, n = 500$ is our final choice of RF's hyper-parameters.
Then the fine-tuned model is used to make predictions on test data. Following figure is the confusion matrix towards all labels:
And the average cost-sensitive measure is:

RF	precision	recall	f1-score
Average	0.48	0.49	0.48

Table 5: Cost-Sensitive Measure of RF

The experiments in Spark with 5 executor and 4 cores for each executor have similar performance as on local, but runtime generally decreases by half.

5.1.3 Results of CNN

Figure 3 below shows the loss and accuracy on train and test set during the 150 epochs training process. We found that both test loss and test accuracy improved comparatively very fast over the first

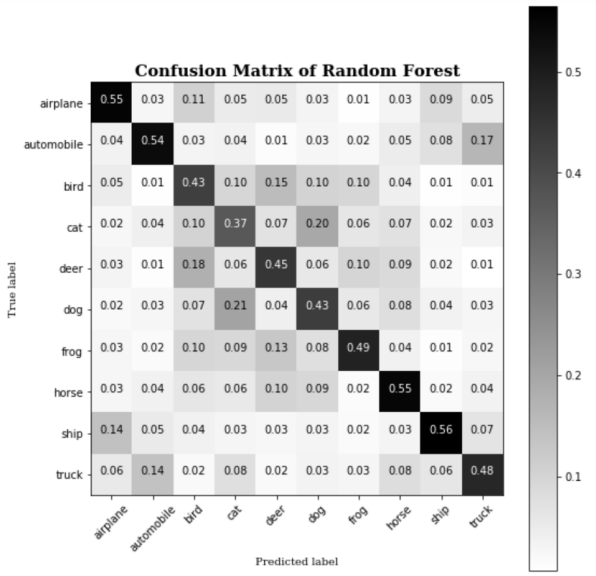

Figure 3: Confusion Matrix of RF

60 training epochs, but has almost no improvement after 80 epochs. However, the train loss and train accuracy still improving after 80 epochs. This indicates the trend of over-fitting to the training data set. We thus can conclude that 80 to 90 epochs might be the optimal choice for training our CNN.

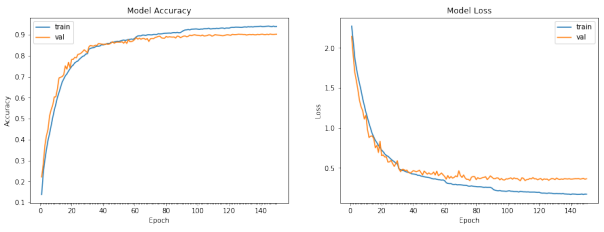

Figure 4: Accuracy and Loss of the CNN during 150 epochs of training

We can see that the accuracy of CNN on test has largely outperformed the other two models used in this report, which also proved the CNN's remarkable performance on conducting image recognition tasks.
We summarized the results and formalized the confusion matrix from the Convolutional Neural Network (11 convolutional layers, 1 global average pooling layer, 0 fully connected layer, with the data augmentation) and shown as follows:

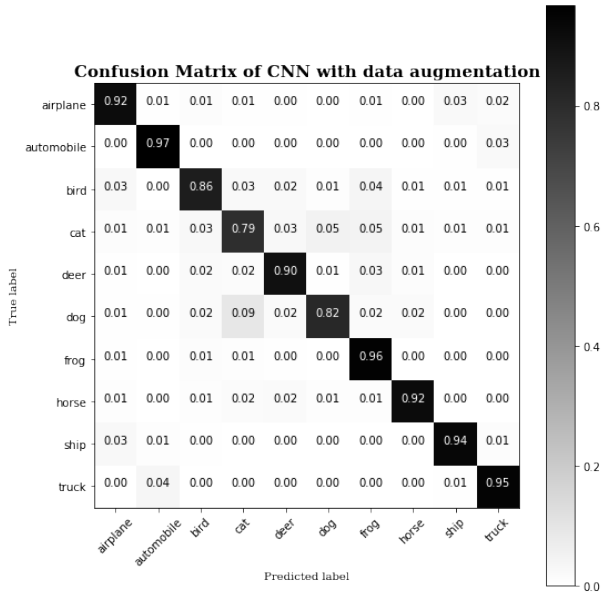

Figure 5: Confusion Matrix of CNN

We found that most classes have very high accuracy, except the classes cats and dogs. Same as the results from KNN, these two classes are likely to be misclassified and this may caused by the similarity between these two classes. Additionally, it gives us the spaces and direction that leads us to improve our model further.

The cost-sensitive measure plan for the Convolutional Neural Network are showed as follows:

CNN	precision	recall	f1-score
Average	0.90	0.90	0.90

Table 6: Cost-Sensitive Measure of CNN

5.2 Runtime, Software and Hardware Specifications

KNN and RF are running on the hardware and software specifications as follows:

Processor	2.7GHz Intel Core i5
Memory	8GB 1867 MHz DDR3
Graphics Card	Intel Iris Graphics 6100
Storage	256GB SSD
Platform	Anaconda
Application	Jupyter Notebook
Interpreter	Python 3.6.0

RF is also experimented in parallel via Spark and Yarn on SIT lab cluster with 30 identical virtual machines (1 master node and 29 slave nodes). The specifications of cluster machines and Spark setting as follows:

Nodes Capacity	4 cores 8GB memory 250GB storage
Executor Number	5
Executor-Core Number	4
Scheduler Mode	FIFO
Driver Memory	1GB
Executor Memory	2GB

CNN are running in a virtual machine hosted by Google, the hardware and software specifications as follows:

Machine Type	Intel Haswell 8 vGPUs
Memory	52GB
Graphics Card	Nvidia Tesla P100
System	Ubuntu Linux
Storage	512GB SSD
Platform	Anaconda
Application	Jupyter Notebook
Interpreter	Python 3.6.0

Modeling the training dataset with 50000 instances and test dataset with 10000 instances, the execution status is as follows:

KNN	
File Loading Time	2s
PCA	17s
Training in 10-fold CV	8m35s
Test Prediction	30s

Random Forest	
File Loading Time	2s
PCA	15s
Training in 10-fold CV	30m11s
Test Prediction	1s
Runtime in Spark	15m20s

CNN	
File Loading Time	2s
Training 100 Epochs	41m50s
Test Prediction	1s

5.3 Discussion

In general, CNN outperforms the other two models as in the image classification features are the RGB values of each pixel so its information is kind of lost when applying classifiers with simple hypothesis like kNN and RF.
Recall that the state-of-art classifier for CIFAR-10 data set achieved more than 95% accuracy on test set, which is 5% more than our best model.

6 Conclusion & Future Work

6.1 Conclusions

As the final conclusion, although along with the deep complexity in model structure and large computation runtime, the optimal classifier methods is Convolutional Neural Network with highest 90% accuracy. Future work will include the ensemble methods which combine multiple CNNs and average their results, and this could be one possible way to improve the current accuracy. Another way is to train another classifier to identify the animals and non-animals first, then classifier the subclass of different animals.

6.2 Future Work

Our CNN could be improved by the following techniques:

- Implement the pre-processing methods such as whitening introduced by Krizhevsky[14] and Global Contrast Normalization suggested by Andrew[4].
- Using ensemble methods, which trained several CNNs in different architectures, and make the prediction together.

References

[1] Yehya Abouelnaga, Ola S Ali, Hager Rady, and Mohamed Moustafa. Cifar-10: Knn-based ensemble of classifiers. In *Computational Science and Computational Intelligence (CSCI), 2016 International Conference on*, pages 1192–1195. IEEE, 2016.

[2] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007.

[3] Yu-Chi Chung, I-Fang Su, Chiang Lee, and Pei-Chi Liu. Multiple k nearest neighbor search. *World Wide Web*, 20(2):371–398, 2017.

[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[5] Xinyang Deng, Qi Liu, Yong Deng, and Sankaran Mahadevan. An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340:250–261, 2016.

[6] Peter Eckersley and Yomna Nasser. Ai progress measurement. 2017.

[7] Li Feifei, Karpathy Andrej, and Justin Johnson. Cs231n: Convolutional neural networks for visual recognition. *Stanford Tutorials*, 2018.

[8] E Fix and J Hodges. An important contribution to nonparametric discriminant analysis and density estimation. *International Statistical Review*, 3(57):233–238, 1951.

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[10] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[11] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[12] DS Guru, YH Sharath, and S Manjunath. Texture features and knn in classification of flower images. *IJCA, Special Issue on RTIPPR (1)*, pages 21–29, 2010.

[13] Tin Kam Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.

[14] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[15] Alex Krizhevsky and Geoffrey Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40:7, 2010.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[18] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[19] Loonycorn and Inc Total Training. *Machine learning: decision trees and random forests.* Total Training, 2017.

[20] D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870, 2007.

[21] Dengsheng Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870, 2007.

[22] Hao Zhang Alexander C Berg Michael and Maire Jitendra Malik. Svmknn: Discriminative nearest neighbor classification for visual category recognition. *Computer Science Division, EECS Department Univ. of California, Berkeley, CA*, 94720, 2007.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[25] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[26] Baoxun Xu, Yunming Ye, and Lei Nie. An improved random forest classifier for image classification. pages 795–800. IEEE, 2012.

[27] Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, 2013.

Appendix A

Code Usage Instructions

- Execute KNN.ipynb, RF.ipynb, and CNN.ipytb under the sub-folder of zip file with Python 3 interpreter.(Our group use Anaconda - Jupyter Notebook).
- The CNN.ipynb is better to be run on GPU (otherwise it may took days to train the model). Or Instructor can directly use our trained weights and saved CNN structured in our folder namely 'COMP5318_CNN_structure_keras' and 'COMP5318_CNN_trained_weights', and make sure the tensorflow and keras package installed on computer.

Appendix B

Members Contribution

Student name	**Participated**	**Agree to share**
Ye Yuan	Yes	Yes
Handuo Mei	Yes	Yes
Ruoyu Zhuang	Yes	Yes