IMA 7505 - Scientific Project
Report for Imagine Classification by using Machine Learning Methods

Xuefeng Wei
Institut Polytechnique de Paris
xuefeng.wei@ip-paris.fr

Abstract

Traditional machine learning models such as KNN is deployed on this dataset to test their performance. Deep learning models such as CNN were deployed on this dataset. For the CNN models, better results were obtained by changing the model structure and adding more layers for iteration. In this paper, five CNNs were proposed and built from scratch and they were tested with accuracies of 0.68, 0.77, 0.79, 0.81 and 0.82, respectively. Transfer learning was deployed on this dataset to compare the models proposed by the author, and the models selected for transfer learning in this project were DenseNet121, InceptionV3, and MobileNetV3. The test accuracy for transfer learning on Cifar-10 dataset after loading the weight file of ImageNet were 0.92, 0.89 and 0.89. Finally, error analysis is performed for the CNN5 in this project and future directions for improvement and work are proposed.

Index Terms——Convolution, Neural Network, Transfer Learning, Machine Learning

1. Introduction

The Cifar-10 dataset was chosen for this scientific project, which consistes of 60,000 32x32 colour images in 10 different classes. The 10 classes are: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck. These targets are very common in our daily life, therefore it is meaningful to create and test deep learning algorithm on this dataset for classification.

This means that we are able to use the neural network to train these smaller images quickly and improve our neural network, while the colored images mean that more information about the images is retained, which allows the neural network to take into account the color features of the images when performing feature extraction. The characteristics of these datasets make it possible to build CNN models from scratch and iterate on them, as well as to consider more complex neural network structures for comparison.

In this scientific project, the main part can be divided into three parts, the first part is the deployment of traditional machine learning models on Cifar-10, in the project the authors used KNN model. The second part involves building the CNN model from scratch and improving the accuracy of the model on the test set by changing hyperparameters and adding network layers, batch normalization and dropout layers are added to the CNN model to prevent overfitting of the model, and the impact of adding these network layers on the overall performance is explored and then compared to their performance on the test set.The third part applies transfer learning by deploying models with more complex architectures and better performance on other datasets Cifar-10, DenseNet121, InceptionV3 and MobileNetV3 are applied in this case, and explores the impact of pretraining and no pretraining on transfer learning.The last part is error analysis of the CNN5. In this project, statistical analysis of different classes of classification errors is performed, and error examples are output for targets possessing different error rates to explore the causes of errors.

2 different machine learning model, 5 different structures of convolutional neural networks and 3 different convolutional neural networks for transfer learning were designed and applied to this dataset to compare their performance. The dataset was seperated to 50,000 training data and 10,000 testing data.

1.1. Data Exploration

The data can be found from the URL: https://www.cs.toronto.edu/~kriz/cifar.html. Each class is considered to be completely independent. Figure 1 shows the first 100 items of MNIST which include all the classes:

This dataset has widely used in machine and deep learning research. The similar datasets of Cifar-10 involve Cifar-100, wchih consists of more classes and more images.

1.2. Motivations

By using traditional machine learning algorithms to the currently used transfer learning, the motivation is to compare their performance capabilities, because transfer learn-

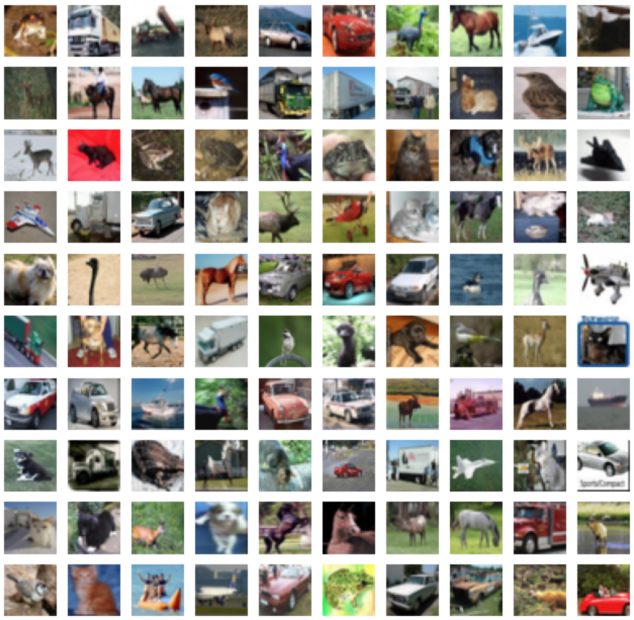

Figure 1. Cifar-10 Sample Images

ing directly uses models that have been built and verified to perform well by other authors, which can greatly reduce our time to develop models. However, it is also extremely important to build CNN models from scratch, so I explored the impact of different hyperparameters on the models by building five CNN models. For the machine learning models, I use KNN because it is classic machine models for classification tasks, which can often be seen in other previous work of the authors. As for Transfer learning, I chose three representative models, the first one is DenseNet121 [3], which proposed by Gao Huang et al., whose model structure is shown in Figure 2. In deep learning networks, the gradient disappearance problem becomes more obvious as the network depth deepens, and the solution is to create short paths between shallow and deep layers, while DenseNet ensures that the network layer and layer DenseNet connects all layers while ensuring maximum information transfer between them.The second transfer learning model is InceptionV3 [1], proposed by google, the design of Inceptionv3 was intended to allow deeper networks while also keeping the number of parameters from growing too large, its architecture is shown in Figure 3,Because of its wide use in different areas of machine learning, I used it for testing on Cifar-10.The last model used for transfer learning is MobileNetV3 [2]because complex and large neural network models are difficult to be applied in real scenarios, such as mobile situations, because submerged devices do not have enough processing power, so small and efficient CNN models are important in these scenarios, so in this project I also tested the performance of MobileNet on Cifar-10 as well,

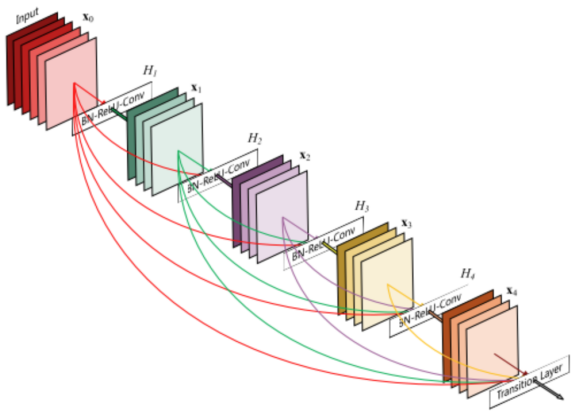

Figure 2. Architecture of DenseNet

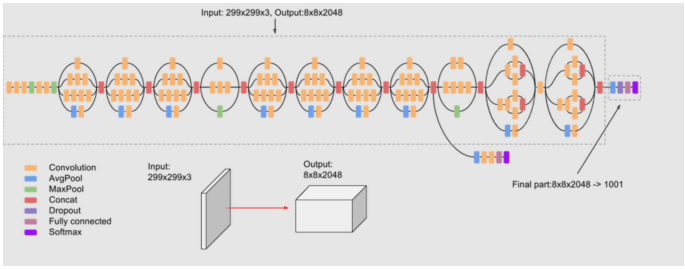

Figure 3. Architecture of InceptionV3

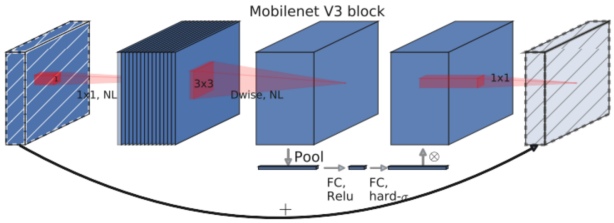

Figure 4. Architecture of MobileNetV3

with the structure shown in the figure 4.

To find which model is more suitable for the Cifar-10 dataset by comparing traditional machine learning models, your own proposed CNN model and models proposed by other authors, and finally perform error analysis on the best and worst model.

2. Method

The first part will introduce the traditional machine learning algorithms used in the project, namely KNN, the second part will introduce the five different CNN architectures proposed by the authors, and the third part will introduce the transfer learning models used.

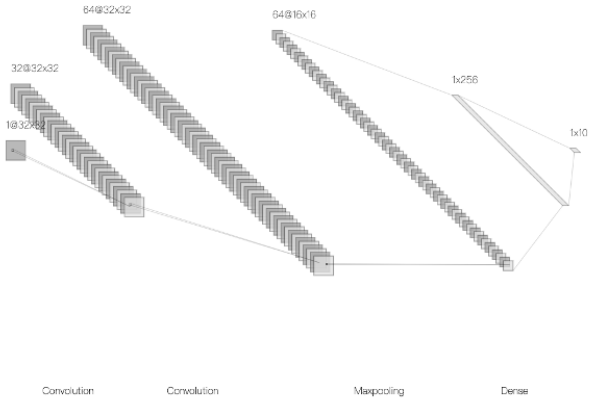

Figure 5. Architecture of CNN1

2.1. Traditional Machine learning Models

In this project, two different traditional machine learning algorithms were applied to the Cifar-10 dataset, the first one is KNN algorithm, i.e. K-nearest neighbour algorithm, because the dataset has too many features, so the authors first used PCA algorithm to grab features from the data, the number of main components set in PCA is 0.95,because for PCA we like to explain between 95-99% of the variance, I set the number of principal components to 95%, i.e. 217 principal components and then used the grabbed main features to classify them using KNN algorithm, and the main hyperparameters of KNN, i.e. the number of K, are discussed.Also, the results of the impact of PCA on classification performance are explored by considering the classification of data without dimensionality reduction by KNN.

2.2. Convolutional Neural Networks

In this project, the author proposed five CNN models with different structures, CNN1,CNN2,CNN3,CNN4 and CNN5, and each CNN structure will be described in detail next.

For CNN1, it is a very simple CNN, it only contains a convolutional layer and a pooling layer,dropout layer(0.25), the activation function is relu, and the classifier is softmax. Because the CNN network is built from scratch, a simple model is chosen to be built at the very beginning, and its network structure diagram is shown in Figure 5.

For CNN2, an additional module is added to CNN1, one module contains 2 convolutional layers, 1 maximum pooling layer and a dropout layer, and the activation function is replaced with LeakyRelu, because compared to Relu as the activation function, LeakyRelu can solve the problem of gradient disappearance in the optimization proces, and its network structure diagram is shown in Figure 6.

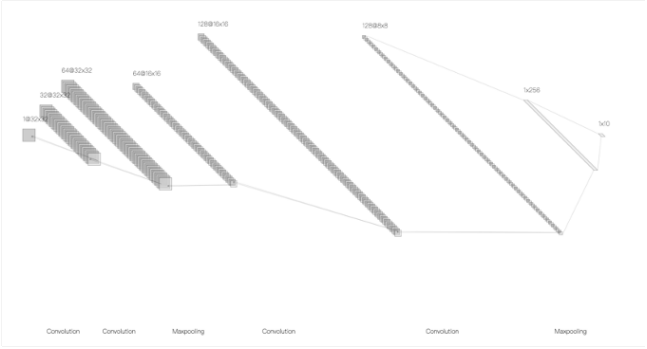

Figure 6. Architecture of CNN2

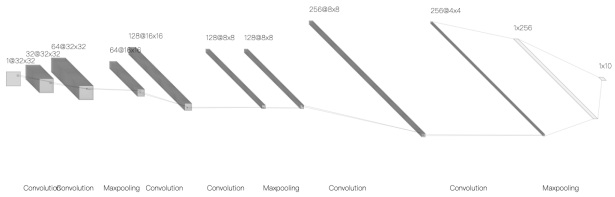

Figure 7. Architecture of CNN3

For CNN3, a module is added to the structure based on CNN2. The main purpose of adding more modules is to bring higher test accuracy through deeper network levels and more parameters, because in neural networks, more parameters and deeper levels mean that more abstract features will be learned by the model to achieve higher accuracy, and its network structure diagram is shown in Figure 7.

By comparing CNN3 with the first two CNN models, more parameters and more complex models are considered, because more convolutional layers are added, so the models are able to learn more abstract and detailed features, but the problem is that as the number of layers increases and the parameters increase, the problem of overfitting arises, which can seriously affect the performance of CNNs, CNN3-1, CNN3-2 are proposed to explore the effect of over fitting and to fix this problem, CNN3 is a model that does not solve the overfitting problem, CNN3-1 adds a dropout layer, and CNN3-2 adds both a dropout layer and a batch normalization layer, the specific structure diagrams of CNN3-1 and CNN3-2 are shown in Figure 8 and Figure 9.

For CNN4, the results are shown in Figure 10 to explore whether deeper and more convolutional layers are effective in improving performance, and the data are augmented to explore the impact on performance while updating the network structure.

Finally, CNN5 is built on the basis of CNN4. All CNN models before that had a convolutional layer kernel size of 3x3, which was inspired by the VGG architecture. Now, in order to consider the impact of a larger convolutional ker-

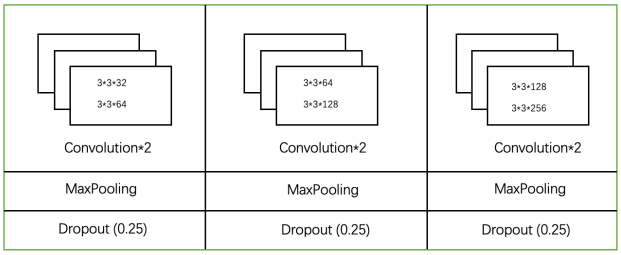

Figure 8. Architecture of CNN3-1

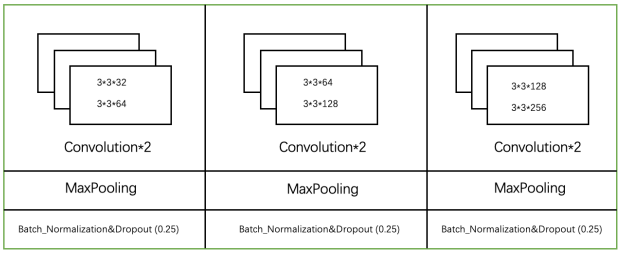

Figure 9. Architecture of CNN3-2

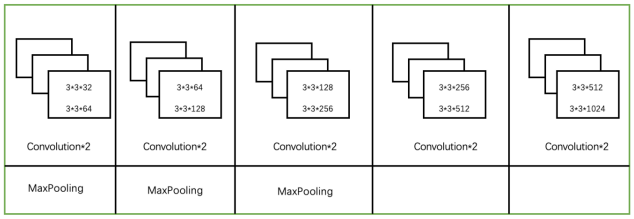

Figure 10. Architecture of CNN4

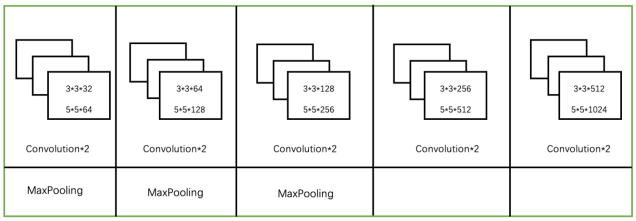

Figure 11. Architecture of CNN5

nel on the CNN, CNN5 is proposed by changing the kernel of the second convolutional layer in each layer to 5x5, and then exploring its impact on the model, whose architecture is shown in Figure 11.

2.3. Transfer learning

In this project, the authors used transfer learning based on DenseNet, Inception and MobileNet, as they all proved to perform excellently on other datasets, and in order for these methods to be applied to the cifar-10 dataset, the models were fine-tuned to accommodate results with only 10 outputs, and their model structures are shown in Figures 12, 13 and 14, respectively.

Input
Feature Extraction (DenseNet121)
Dense Layer (256) Dropout layer (0.3) Batch normalization layer
Dense Layer (128) Dropout layer (0.3) Batch normalization layer
Dense Layer (128) Dropout layer (0.3)
10 outputs in last layer

Figure 12. Architecture of transfer learning on DenseNet121

Input
Feature Extraction (InceptionV3)
Dense Layer (256) Dropout layer (0.3) Batch normalization layer
Dense Layer (128) Dropout layer (0.3) Batch normalization layer
Dense Layer (128) Dropout layer (0.3)
10 outputs in last layer

Figure 13. Architecture of transfer learning on InceptionV3

2.4. Loss Criterion and Optimizer

Cross entropy loss was used to be the loss function for all models in multi-class classification task.

$$L = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

Cross entropy loss was wildly used in classification task since it avoids learning slow-down and helps with the vanishing gradient problem from which deep neural networks suffer, also because softmax was choose as the output layer, and it is a probability distribution.

Input
Feature Extraction (MobileNetV3)
Dense Layer (256) Dropout layer (0.3) Batch normalization layer
Dense Layer (128) Dropout layer (0.3) Batch normalization layer
Dense Layer (128) Dropout layer (0.3)
10 outputs in last layer

Figure 14. Architecture of transfer learning on MobileNetV3

The optimizer that used in the experiments was stochastic gradient descnet optimizer, for SGD, only one training sample is used for each parameter update. Pros: Time saving compared to gradient descent method. Disadvantage: Each iteration is not necessarily the direction of the overall optimization of the model. If the sample is noisy, it is easy to fall into a local optimal solution and converge to an undesirable state.

2.5. Hyper-parameters Setting for training

First of all, for traditional machine learning algorithms, for KNN, the number of principal components is set at 0.95 during the PCA feature capture phase, while the K value for KNN classification is obtained empirically, i.e., the optimal K value is determined by comparing the test accuracy under different K values.

For the convolutional neural network, the epoch is set to 50, because more training times can make the model converge better, the batch size is 128, the chosen optimizer is SGD, the learning speed is 1e-4, the momentum is 0.9, the loss function is crossentropy, and the index to measure the model effect is test accuracy, and the final classifier is softmax.

For transfer learning, the epoch is set to 20, because the transfer learning model has a more complex network structure and more parameters, and a lower number of iterations is chosen because of the computational power. batch size is 128, the optimizer chosen is Adam, the loss function is crossentropy, and the index to measure the effect of the model is test The final classifier is softmax.ImageNet in the choice of weight files, which indicates that these models are pre-trained on ImageNet.

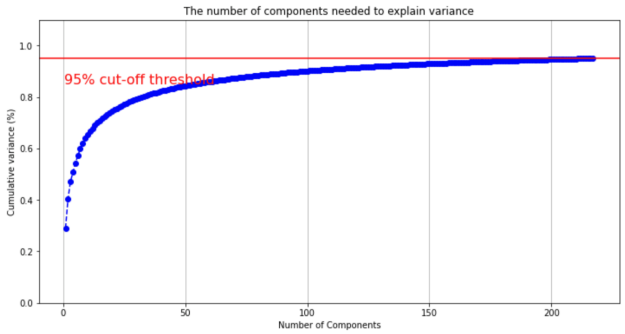

Figure 15. Explainable variance curves in PCA

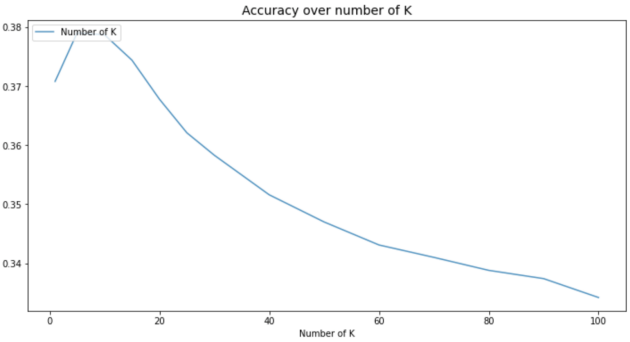

Figure 16. K-value curve after PCA

3. Experimental results

The experimental results are divided into 4 parts, first the experimental results of traditional machine learning algorithms, then the experimental results of CNN models, then the experimental results of transfer learning, and finally the comparison of them.

3.1. Results on KNN

The first experimental results of KNN are shown, and the authors first discuss the results after using PCA to capture the features. For the selection of PCA, the number of main components with an explainable variance of 0.95-0.99 is preferred, so its explainable variance curve is shown in Figure 15.

The authors chose a quantity of 0.95 as the number of principal components, which resulted in 217, and then after choosing the number of principal components, they used KNN to classify them, and to determine the optimal K values, the authors empirically took the K values in the range of 1-100, and the period results are shown in Figure 16.

From Figure 16, it can be seen that the model of KNN reaches the optimal test accuracy when the value of k is equal to 5, and its test accuracy is 0.379.

To explore the effect of the dimensionality reduction operation on KNN, the authors classified the data using KNN

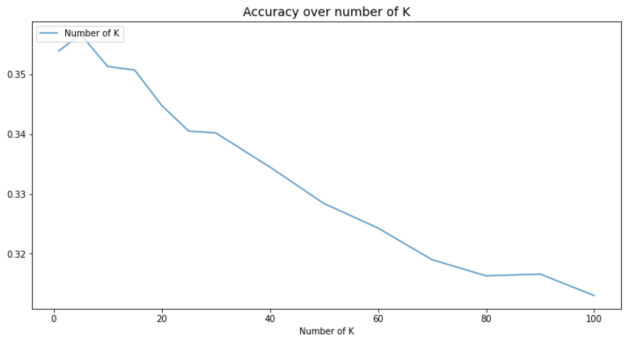

Figure 17. K-value curve without PCA

Model	Test Accuracy
KNN after PCA	**0.3790**
KNN without PCA	0.3567

Table 1. Test Accuracy for KNN

Model	Time Costing on Training
KNN after PCA	**16 Minutes**
KNN without PCA	1 Hour

Table 2. Time Costing on training for KNN

without KNN and also empirically searched for the optimal K value in the range of 1-100, and the results are shown in Figure 17. From Figure 17, it can be seen that the model has the highest accuracy on the test set when the value of k is taken to 5, which is consistent with the KNN classification after PCA operation, but in terms of accuracy value, the classification result after PCA dimensionality reduction is significantly better than the result without dimensionality, which has a test accuracy of 0.3567, Their accuracy rates can be seen in Table 1.

We can also measure the effect of PCA on the training time, and their respective runtime results can be seen in Table 2.Finally, their confusion matrix can be seen in Figures 18 and 19.

3.2. Results on CNN models

In this section, the experimental results of the CNN model pair will be shown, firstly, the learning curves and confusion matrices of CNN1-3 are shown in Figures 19, 20 and 21, and the confusion matrices are shown in Figures 22, 23 and 24.

The result plots from CNN1 to CNN3 show that the model performs better as the number of convolutional layers and the parameters increase. Their accuracy on the test set can be seen in Table 3.

With the increase of network parameters and network levels, the problem of overfitting appears in CNN models.

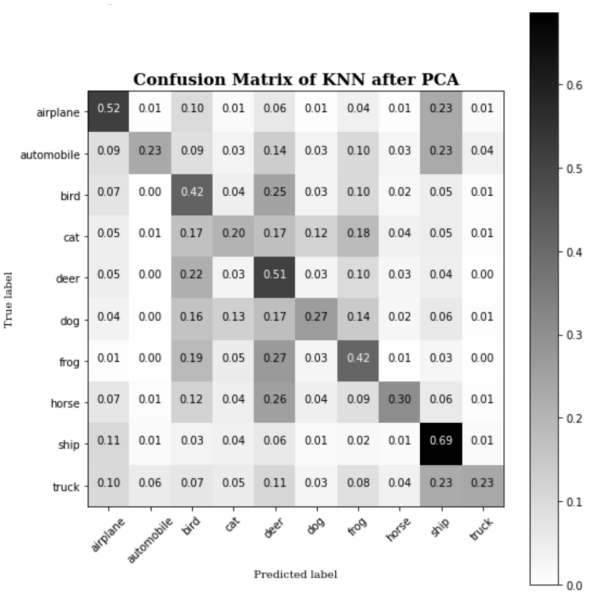

Figure 18. Confusion Matrix for KNN after PCA

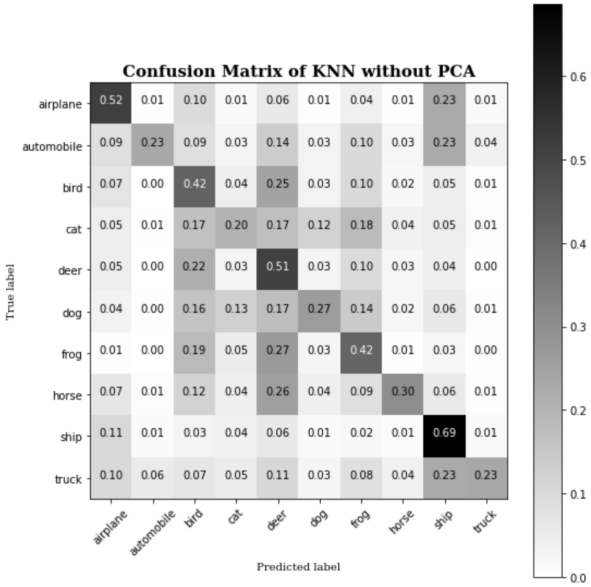

Figure 19. Confusion Matrix for KNN without PCA

Model	Time Accuracy
CNN1	0.6808
CNN2	0.7728
CNN3	**0.7957**

Table 3. Test Accuracy for CNN1, CNN2 and CNN3

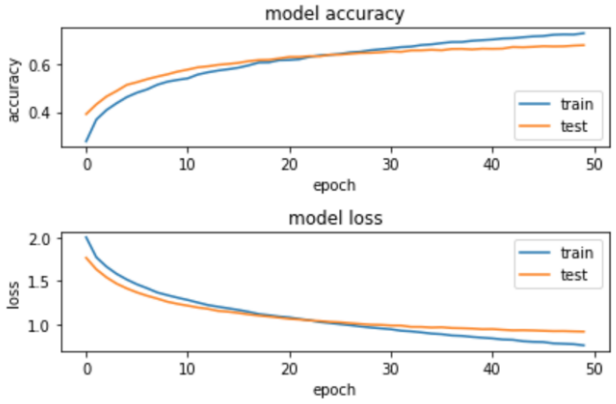

Figure 20. Learning Curve of CNN1

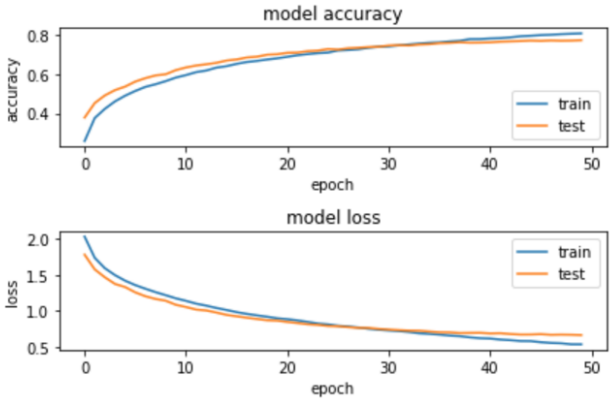

Figure 21. Learning Curve of CNN2

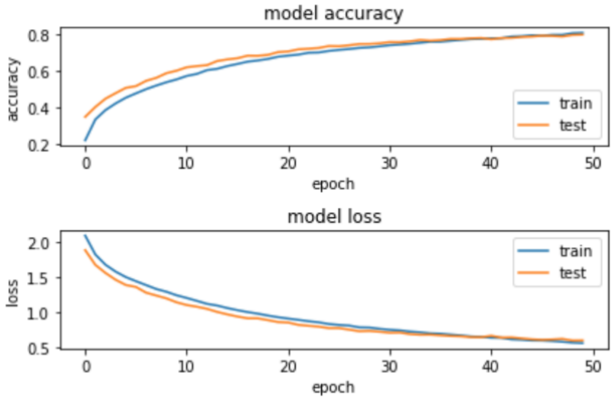

Figure 22. Learning Curve of CNN3

In order to explore dropout and batch normalization layers for overfitting solution, three different CNNs, CNN3-1,CNN3-2 and CNN3-3, are proposed without special layer, with dropout layer and with CNN models with dropout and batch normalization layers. Figure 26, Figure 27 and Figure 28 show their learning curves.

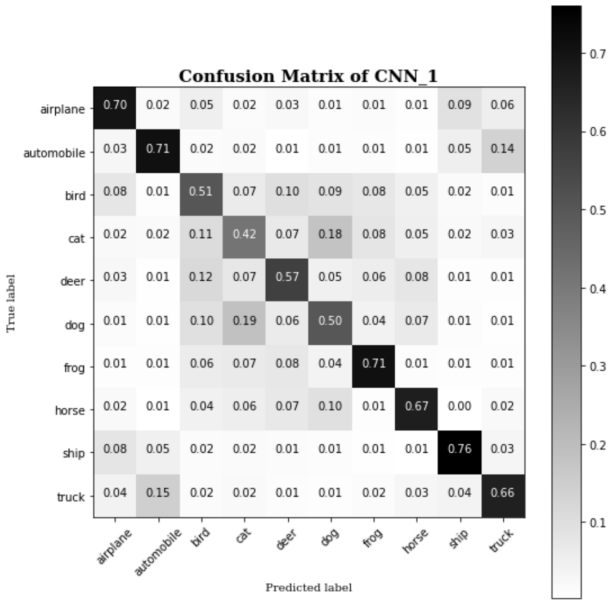

Figure 23. Confusion Matrix of CNN1

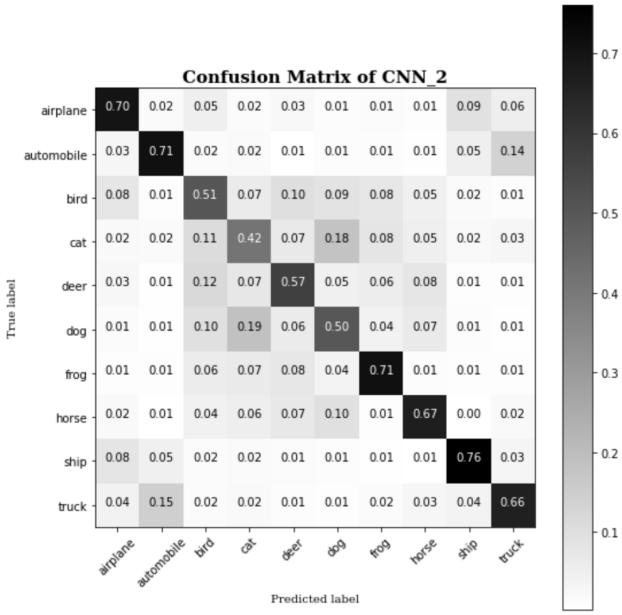

Figure 24. Confusion Matrix of CNN2

From the learning curves of CNN3-1, CNN3-2 and CNN3-3, we know that the CNN models without dropout and batch normalization layers are more prone to overfitting, while the models with both layers show faster con-

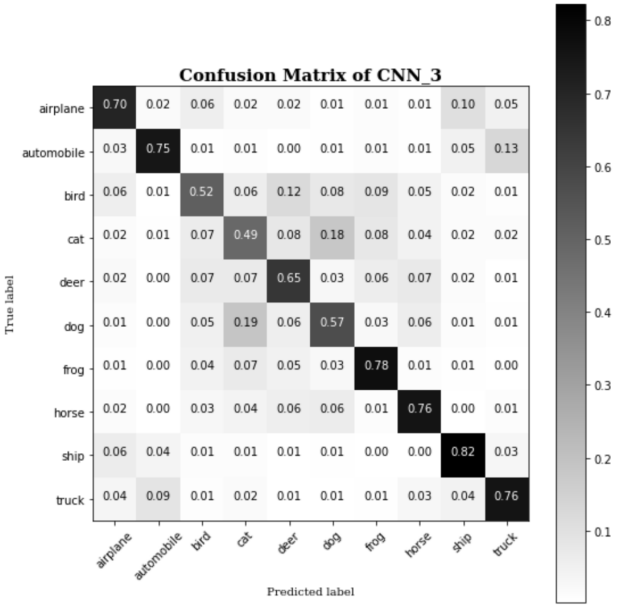

Figure 25. Confusion Matrix of CNN3

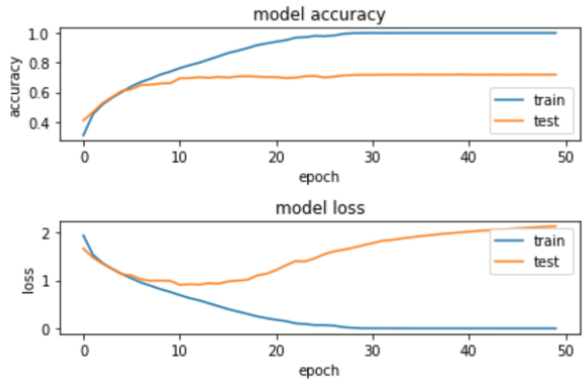

Figure 26. Learning Curve of CNN3-1

Model	Time Accuracy
CNN3-1	0.7187
CNN3-2	0.7957
CNN3-3	**0.8041**

Table 4. Test Accuracy for CNN3-1, CNN3-2 and CNN3-3

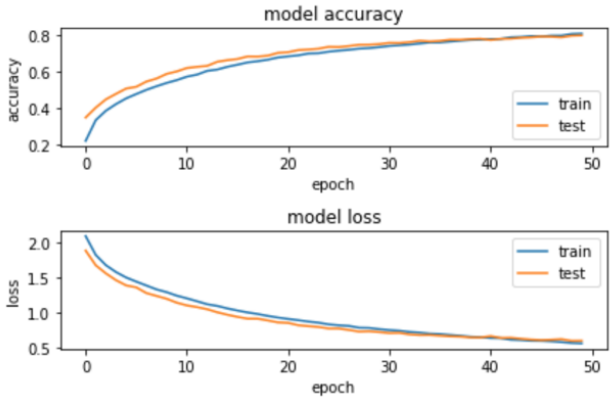

Figure 27. Learning Curve of CNN3-2

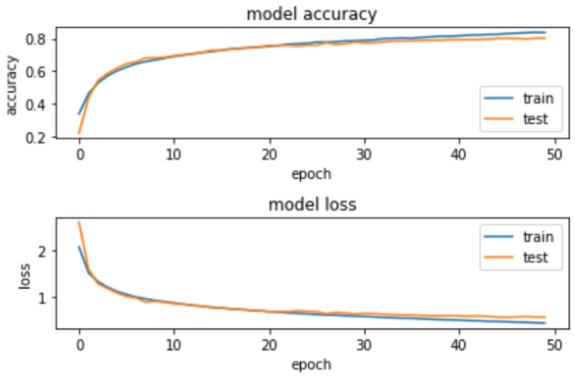

Figure 28. Learning Curve of CNN3-3

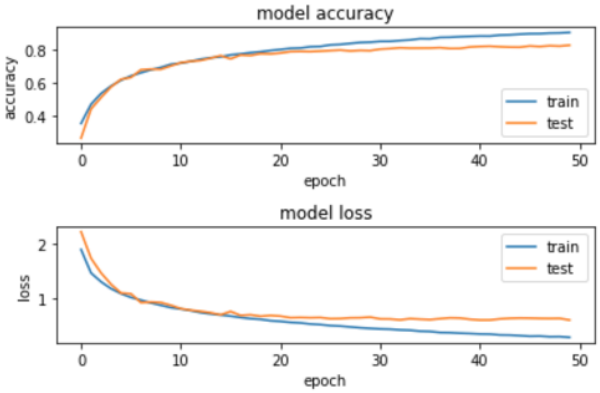

Figure 29. Learning Curve of CNN4

vergence and better accuracy, and their test accuracies are shown in Table 4.

It is also clear from the test accuracy that the dropout and BN layers solve the overfitting problem of CNNs, leading to deeper and more parameterized CNNs becoming possible. The next experimental results are CNN4 and CNN5, which are based on CNN3 with more convolutional layers and deeper, and also consider the effect of larger convolutional kernels on the model, and their learning curves are shown in Figures 29 and 30, and the confusion matrix is shown in Figures 31 and 32.The test accuracies of CNN4 and CNN5 are then shown in Table 5.

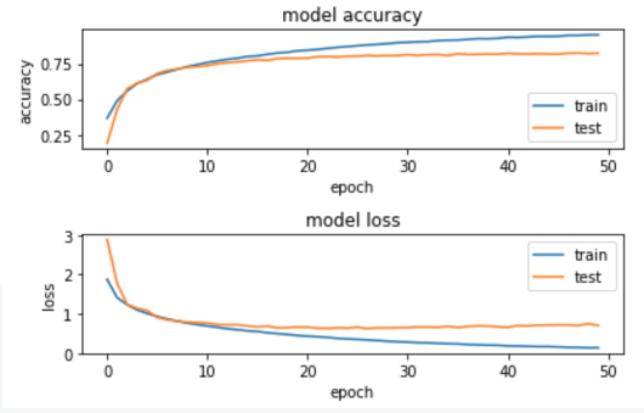

Figure 30. Learning Curve of CNN5

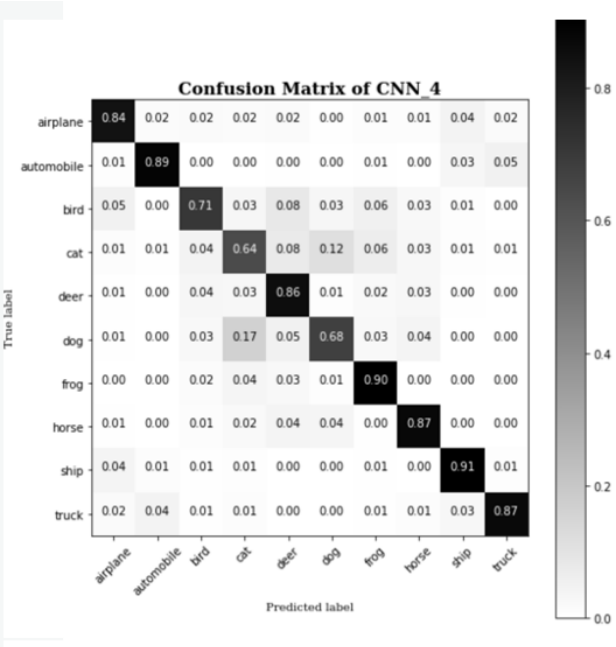

Figure 31. Confusion Matrix of CNN4

Model	Time Accuracy
CNN4	0.8168
CNN5	0.8204

Table 5. Test Accuracy for CNN4, CNN5

From the results in Table 5, more convolutional layers improve the performance of the model, but the improvement effect is small, along with the increase is the training time of the model, the training time of CNN5 is 1026s, while CNN4's is 402s, while CNN3 can be less, 311s, so what is known is that as the model becomes more complex, the time cost rises, and the improvement is getting smaller.

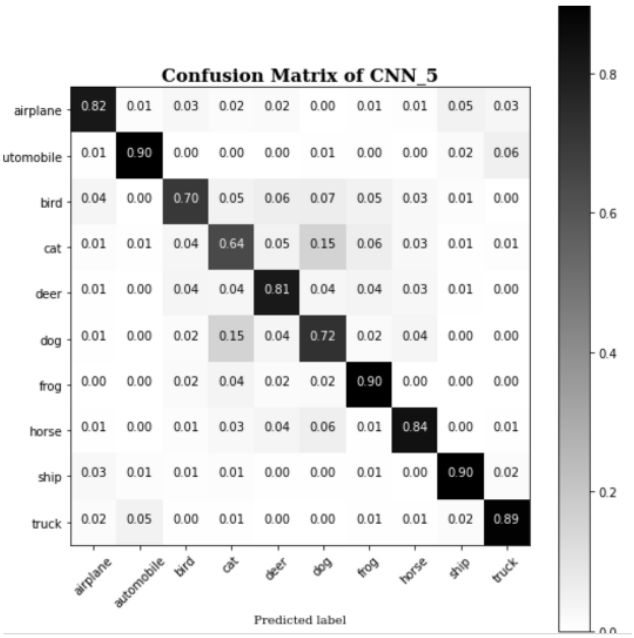

Figure 32. Confusion Matrix of CNN5

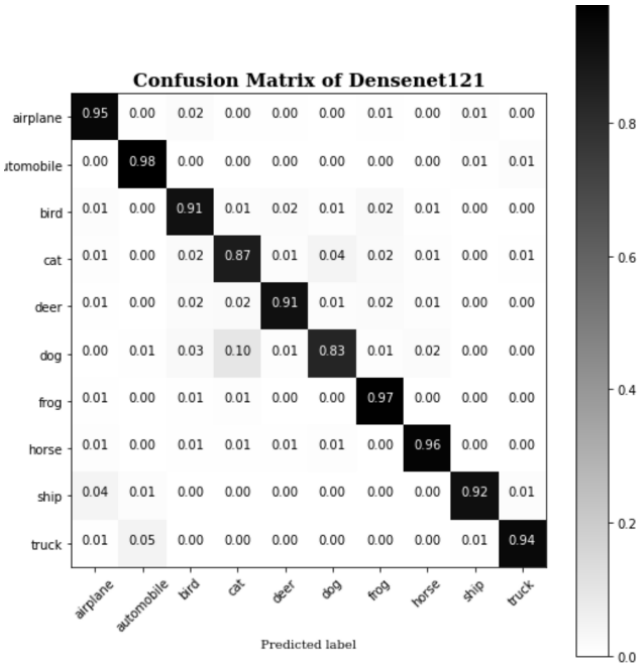

Figure 33. Confusion Matrix for DenseNet121

3.3. Results on Transfer Learning

First are the experimental results based on pre-trained transfer learning, and their confusion matrices are shown in Figure 33, Figure 34 and Figure 35. Their test accuracies are shown in Table 6. Finally, to explore the effect

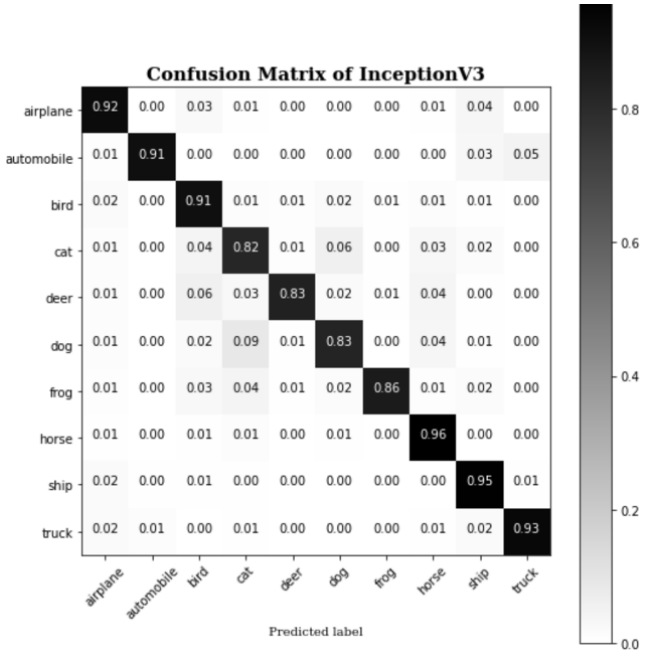

Figure 34. Confusion Matrix for InceptionV3

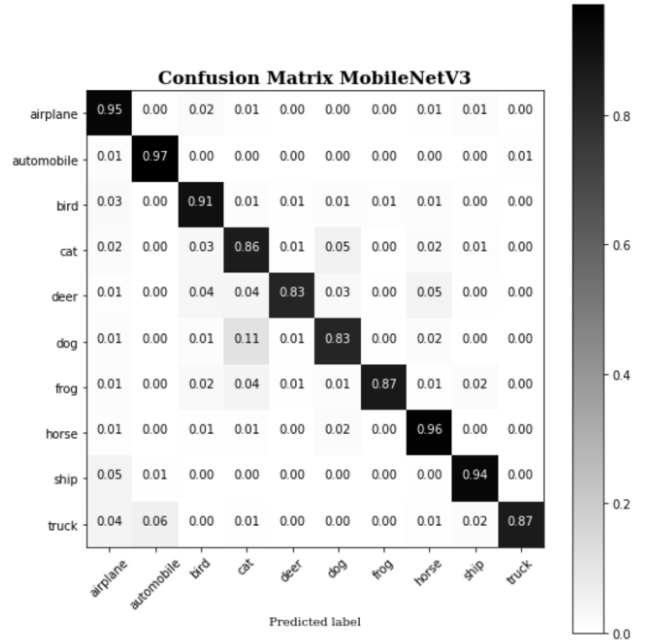

Figure 35. Confusion Matrix for MobileNetV3

of pre-training on the performance of transfer learning on the cifar-10 dataset, author used InceptionV3 for testing and compared the test accuracy with the previous InceptionV3 without using the weight file, as shown in Table 7. From the results in Table 7, it is clear that pre-training can improve the performance of transfer learning because the ImageNet dataset contains many targets that are also available in the Cifar-10 dataset, and the more datasets there are, the better the performance of the CNN will be, which naturally improves its performance.

Model	Time Accuracy
DenseNet121	**0.9236**
InceptionV3	0.8935
MobileNetV3	0.8972

Table 6. Test Accuracy for Pre-trained transfer learning

Model	Time Accuracy
InceptionV3 pretrained	**0.8935**
InceptionV3 not pretrained	0.8441

Table 7. Test Accuracy for InceptionV3

Model	Time Accuracy
InceptionV3 pretrained	**0.8935**
InceptionV3 not pretrained	0.8441

Table 8. Test Accuracy for InceptionV3

Models	Test Accuracy
KNN after PCA	0.3790
CNN5	0.8204
DenseNet121	**0.9236**

Table 9. Test Accuracy for the best model from different algorithm

3.4. Performance Comparison

In this section all the results will be summarized and compared and the results are shown in Table 8.The models with the best results for each algorithm will be screened for comparison

Table 8 represents the best models of the three different methods, and from the results, the method based on transfer learning obtained the best results with an accuracy of 0.9236, while the method based on traditional machine learning obtained the worst results, which also shows that deep learning is far better than traditional machine learning methods with sufficient computer power today. And by comparing the CNN and transfer learning proposed by the authors we can see that there is still a large gap, which is the higher the accuracy is also more difficult to improve, this is because there are targets with more similar features and more uncommon object angles. The more complex and carefully designed model results in better performance.

Class	Error Rate
Frog,Ship,Truck	0-0.1
Automobile,Deer,Horse	0.1-0.2
Airplane,Bird,Dog	0.2-0.3
Cat	0.3-0.4

Table 10. Error rate in CNN5

Class	Error Rate
Dog	0.3-0.4
Deer,Frog,Bird	0.1-0.2
Airplane,Automobile,Horse,Ship,Truck	0-0.1

Table 11. Error rate in CNN5 for Cat

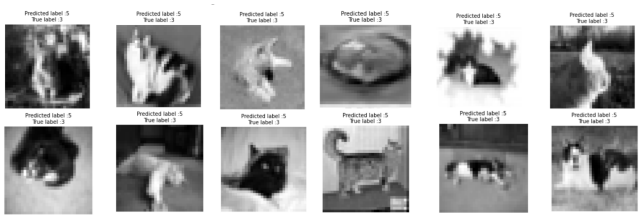

Figure 36. Error images for cat to dog in CNN5

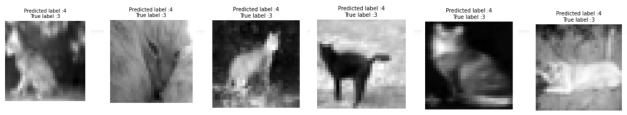

Figure 37. Error images for cat to deer in CNN5

4. Error Analysis

For CNN5, the authors performed statistics on misclassified targets, and the results are shown in Table 10.

From Table 10, it can be seen that the errors of CNN5 are clearly divided into four classes, and the authors here choose representative categories for error analysis, namely Cat. For the category of cats, the authors counted the error rate of cats being misclassified to other species, as shown in Table 11. For the category of cats, the authors calculated the error rate of cats being misclassified to other species, as shown in Table 11, from which it can be seen that cats are most often classified to dogs, followed by deer, frogs, and finally airplanes, cars, etc.So, the authors output some wrong images for analysis, starting with the wrong image of a cat being classified as a dog, as shown in Figure 35. As can be seen in Figure 35, because dogs and cats have similar morphological characteristics, the model will easily misclassify cats and dogs, and some images are very confusing even in human eyes. Then the cat was misclassified as a deer,The error image is shown by Figure 37. As can

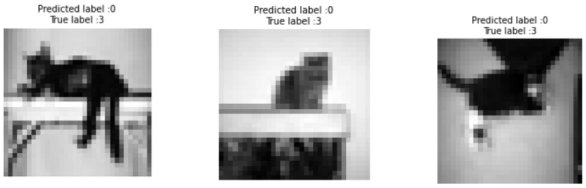

Figure 38. Error images for cat to airplane in CNN5

Class	Error Rate
Automobile	0.3-0.4
Ship	0.2-0.3
Airplane,Bird,Cat,Deer,Dog,Frog,Horse	0-0.1

Table 12. Error rate in CNN5 for Truck

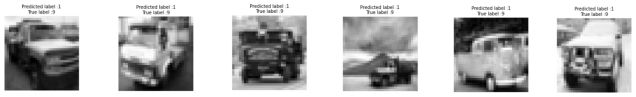

Figure 39. Error images for truck to automobile in CNN5

be seen in Figure 36, in the case where cats are classified as deer, they commonly have similar phase characteristics to deer, i.e. standing on all fours, or creeping, which is related to deer habits, so the model classifies them as deer. The last one shows the cat being classified as an airplane, and its error image is shown by Figure 38. The last one is a cat classified as an airplane, and its wrong image is shown in Figure 38, from which it can be seen that for the first two misclassified images there is a common point that these cats are on a mid-air platform, which is similar to the airplane point feature, while the last one is because it happens that the cat point legs and tail form an angle similar to the airplane wing point, so it is mistakenly considered as an airplane by the model.

Then the author chooses the truck for the error analysis, because compared to the cat, the truck is a human-made object and has different morphological characteristics from the animal,The error rate can be seen in Table 12. From Table 12, it can be seen that trucks are more likely to be confused with cars, which is consistent with our perception because they have many overlapping features, while the second item is boats, so it can be concluded that man-made objects are more likely to be confused by the model as similar man-made object.Specific examples of errors can be seen in Figure 39, Figure 40 and Figure 41. The truck error example shows that even for humans, some pictures of cars can easily be mistaken for trucks, while for trucks mistaken for ships because the angle of the shot caused a similarity with the shape of the truck leading to the model's error of judg-

Figure 40. Error images for truck to ship in CNN5

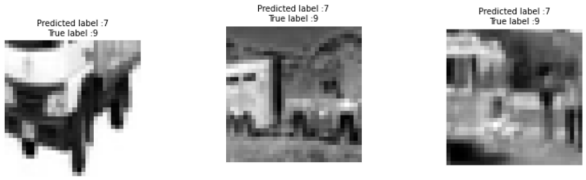

Figure 41. Error images for truck to horse in CNN5

ment, and finally the truck is mistaken for a horse, where it is difficult for humans to explain why such a phenomenon is caused, and in the future it is necessary to answer this question with the help of interpretable artificial intelligence technology.

5. Final Conclusion and Future Work

In general, in this project traditional machine learning algorithms were first used to classify the KNN dataset, specifically KNN. In the KNN classification experiments, the effect of using PCA to capture features on the KNN model was discussed, and it turned out that the model with PCA for dimensionality reduction had a better performance both in terms of training time spent and test accuracy performance, and the optimal K value was obtained empirically, fixing the most significant hyperparameters of KNN. In the experimental part of building a CNN model from scratch, a total of 5 models were proposed and the performance was enhanced as the model was iterated, but the problem for CNN was overfitting, so to address and explore the impact of overfitting on CNN, dropout and batch normalization layers were added to the CNN model to discuss the fix and impact on overfitting, and finally the data showed that the CNN model with both of these layers had better resistance to overfitting and the final test accuracy was better than the CNN model without the addition of dropout and BN layers. In the transfer learning section, migration learning models based on DenseNet, inception and MobileNet were deployed and fine-tuned to fit the Cifar-10 dataset, first using the models pre-trained on ImageNet, and finally achieving 0.9236, 0.8935 and 0.8972 test accuracies, respectively. To explore the effect of pretraining on transfer learning, the test accuracy was compared with and without loading the ImageNet weight file based on InceptionV3, and finally the pretrained model was shown to outperform the model without pretraining. Finally, through error analysis, we know that the model will easily confuse animals with animals and man-made objects with man-made objects because they have similar morphological characteristics, while for animals, the pose of different animals, the angle of the picture and the environment where the objects are located will affect the model's judgment, where confusion of animals is most likely to occur because their physique and position are more variable.For future work, to improve the accuracy, consider introducing more advanced models, such as models based on attention mechanisms or self-supervised models, and consider interpreting the models and introducing interpretable AI techniques, such as SHAP.

References

[1] Szegedy Christian, Liu Wei, Jia Yangqing, Sermanet Pierre, Reed Scott, Anguelov Dragomir, Erhan Dumitru, Vanhoucke Vincent, and Rabinovich Andrew. Going deeper with convolutions. *CVPR*, 2015. 2

[2] Andrew G. Howard, Zhu Menglong, Chen Bo, Kalenichenko Dmitry, Wang Weijun, Weyand Tobias, Andreetto Marco, and Adam Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv 1704.04861*, 2017. 2

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *CVPR*, pages 4700–4708, 2017. 2