Image classification evaluation between CNN and fully connected neural network

Jiawei Liu

Research school of Computer Science,
Australia National University,
u7139102@anu.edu.au

Abstract. This paper will extents my previous work which gives the vehicle fine-grained results by using multi-task learning method. Besides, this paper will examine whether the convolutional neural network (CNN) can obtain a better performance on image-processing tasks than a normal fully connected network (FC network), and whether a better fitting can be obtained by stacking the convolutional layers. AlexNet is selected as the deeper layer model. The result of the experiment shows that the CNN could achieve higher accuracy, less memory consumption and faster converging speed in image-processing tasks than a normal FC network. AlexNet could achieve faster converging and training speed than the shallow CNN. Furthermore, some additional optimizing method such as neuron pruning, batch normalization is used to get better performance.

Keywords: Convolutional neuron network, image classification, neuron pruning.

1 Introduction

Vehicle recognition and fine-grained classification has become a crucial topic in modern intelligent city. The intelligent machine needs be able to identify and give the classify the specific vehicle. To solve this problem, there have been much previous work published. S. Kaewkamnerd and et al[1] tries to use the embedded wireless magnetic sensor for real-time vehicle classification scenario. WM. Liu and et al[2] designs an automatic vehicle classification instrument for high way toll collection system based on multiple sensor information fusion. SY Cheung and et al[3] provide a method that uses magnetic senor to monitor the temporary traffic measurement.

The neuron network (NN) is a type of the deep learning model and has recently become more favorable in AI tasks than traditional machine learning method such as decision tree, SVM. The NN based algorithms have achieved outstanding results on many fields, such as facial recognition[4], real-time object detection[5], machine translation[6] and etc. For most of the image processing tasks, convolutional neural network (CNN) will usually be capable of. Many robust CNN based network have been developed, like YOLOv3[5], faster R-CNN[7] and deep ResNet[8].

This paper plans to prove that CNN will achieve a better fitting to the image classification tasks than the FC net. Furthermore, this paper will also look at whether by stacking more convolution layers, the network could learn the complex features better than a less convolution layers network and reduce the overfitting. The evidence will be provided by comparing the models' loss and multi-task learning accuracy. Three different models are tested in the experiment, which are a fully connected neural network (FC net), a CNN with two convolution layers and linear multi-task classifier layer and the AlexNet[9] with multi-task classifier. We will evaluate the performance of those models by comparing their loss and accuracy converging pattern and training speed. Gedeon and et al[10] introduces that pruning network by distinctiveness could reduce the overfitting in network training, therefore it will be used to adjust the neurons during training to reduce overfit. The experiment result shows that CNN based model with deeper convolutional layers could boost the converging speed, especially in learning complex non-linear features. For the pruning method, it does not give much contribution for the training. The dataset is consist of vehicles which are generated from VehicleX[11] which is an engine to synthesize the real-world vehicles. Additionally, rather than purely train the network, we tuned the number of layers and number of hidden neurons and applied other optimization methods to investigate the better performance.

The contents of the paper are organized into this structure: section 2 will introduce the experiment methods of this paper, including the dataset structure and information preprocessing, structure of FC net, simple CNN and the AlexNet and the neuron pruning algorithm. In addition, section 2 will also introduce the evaluation metrics. Section 3 will show all the experiment results and the discussion about the results. Section 4 will conclude the paper and give future work

2 Methodology

2.1 VehicleX dataset, data preprocessing

VehicleX[11] is an open source image engine which is implemented by Unity. It could simulate and generate virtual 3D real-world vehicles. Its dataset contains 1,362 various simulated vehicles (1,362 vehicleID). The vehicles have 12 colors

and 11 types. In this paper, we use the images that are generated by this engine. The generated dataset contains 45,438 images for training, 14,936 images for validation and 15,412 images for testing. The data distribution is shown in figure 1. Each image is an RGB jpg with dimension of $3 \times 256 \times 256$. There will be some data augments applied for the images includes random rotation and random horizon flip, then the image will be converted to tensor and put into the network.

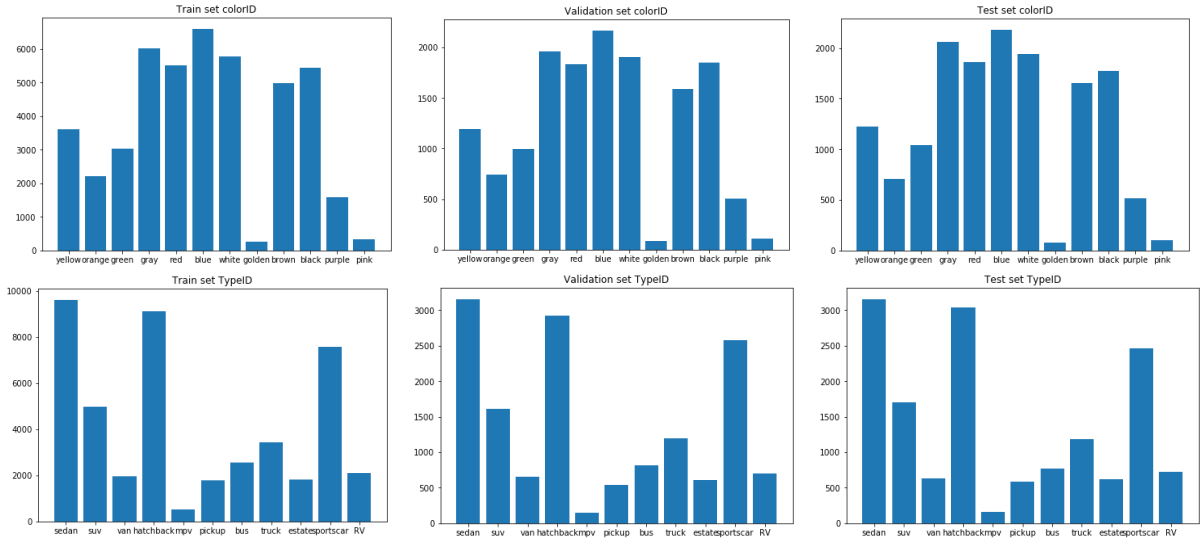

Fig. 1 Data distribution for the dataset

The labels of each image can be separated into two stages. First stage contains the computer vision related labels (camera height, camera distance, camera direction, light intensity, light direction, vehicle orientation). The second stage contains vehicle property related labels (colorID, typeID, vehicleID).

For this experiment, only vehicle property related labels are used, and each ID will be given a classification task. The expected output of the model are the color and type of the input vehicle image and its corresponding vehicleID. The colorID and typeD are relatively simple classification tasks, since the total number of classes is not large (12 and 11 respectively). However, vehicleID is a complex task, because of the large number of classes (total 1,362).

2.2 FC Network

First, we provide a fully connected network with multi-task learning functionality as the benchmark. The architecture of the fully connected network is shown in figure 2. Due to the extreme large memory consumption of the image processing, the experiment device does not have so many memories to afford for the direct input of the image. Therefore, we add a 2D max pooling layer to down sample the input image before inputting it to the first layer. There are three classification tasks for the model, which are the vehicleID classification (1362 classes), colorID classification (12 classes) and typeID classification (11 types).

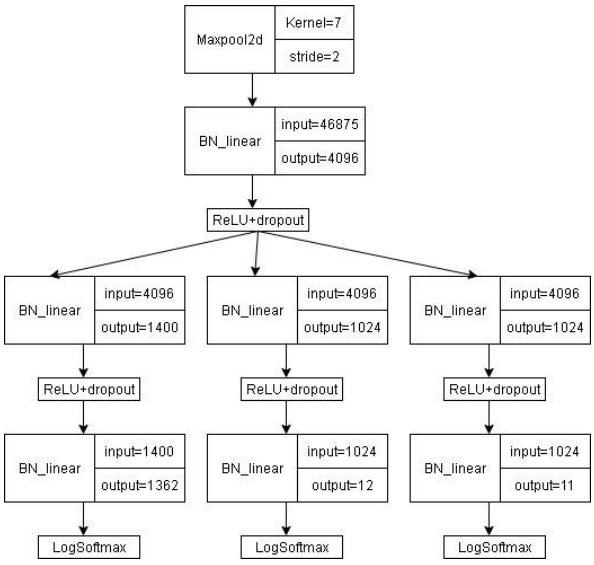

Fig. 2 Structure of the simple FC net. BN_linear means do batch normalization after the linear layer.

The max pooling will down sample the image according to the pooling kernel and the stride (figure 3). The output value of the max pooling layer would be the maximum value inside the pooling kernel. The output size of the max pooling can be calculated by Eq. 1. Due to the down sampling functionality, the spatial complex in training will be reduced.

$$Dimension\ reduced = \frac{\text{input size} - \text{kernel size}}{\text{stride}} \tag{1}$$

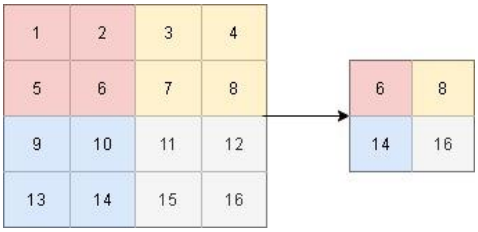

Fig. 3 Example of 3X3 max pooling output

We use the ReLU for the hidden layer activation function and a dropout layer with rate of 0.5. Since this is a multi-class classification task, the loss function will be calculated by using the categorical cross-entropy loss (Eq. 3) with log softmax function (Eq. 2). Where $\hat{y}$ and y denotes the target and prediction

$$\text{LogSoftmax}(\text{x}_\text{i}) = \log\left(\frac{\exp(\text{x}_\text{i})}{\sum_\text{j}\exp(\text{x}_\text{j})}\right) \tag{2}$$

$$l(\hat{y}, y) = -\sum_{k}^{k} y^{(k)} \log\left(\hat{y}^{(k)}\right) \tag{3}$$

2.3 Simple CNN

The second tested model is a CNN with two convolutional layers (Figure 4). The input will be convolved and pooled twice, then be flatten to a 1D tensor before getting into the hidden layers. The hidden layer classifier maintains the same setting as the FC net hidden layers, including the activation function, dropout rate and number of output classes.

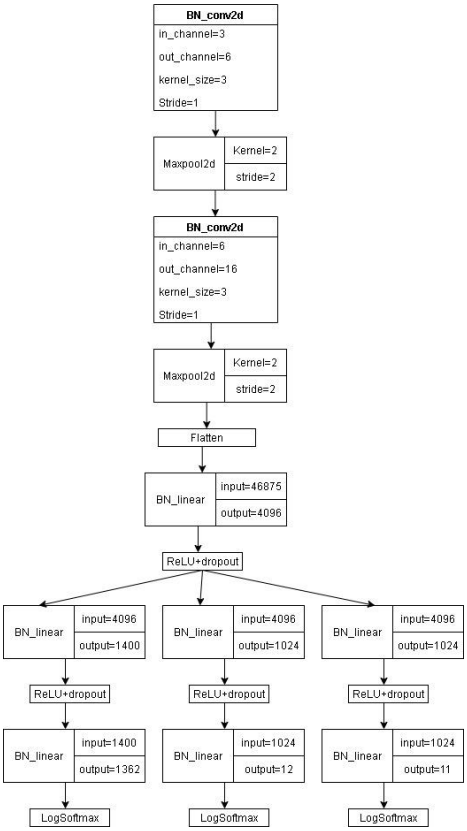

Fig. 4 Structure of the simple CNN. BN_conv2d means batch normalize after the convolution, and BN_linear means batch normalize after the fully connected layer.

CNN is a deep learning model that can take an input image and assign importance to various features or objects in the image and can differentiate one from another. The spatial and temporal dependencies in the image could be successfully captured by the CNN through the application of sliding kernel filters. Each of the filter has a specific set of neurons, then the specific area of input image will be convolved with those filter kernels to produce a feature map. The feature map convolution can be calculated through Eq. 4.

$$feature\ map = input \otimes kernel = \sum_{y=0}^{columns} \left(\sum_{x=0}^{rows} input(x-a, y-b) kernel(x,y) \right) \tag{4}$$

Since the kernel is sliding when calculating the feature maps, the output will be sliding dot product between the input and the kernel, a and b denotes the sliding steps in column and row direction. The convolution between the kernels and input will produce a series of neurons in an activation map that detect features in different regions of the input data. The pooling method is same as the 2D max pooling that was discussed in section 2.2 Eq. 1.

Finally, the input to fully connected layer from the convolution layer is width and height of the image that reduced by the pooling layers multiply the number of feature maps (out_channels).

With the assistant of the variety of dimensionality reduction techniques in the CNN, the computational load in the FC layers will be significantly reduced compares to the normal NN. Besides, the kernel method would also allow the network to study image's detail features locally and globally depends on the kernel.

2.4 AlexNet

To test how the different architecture and depth in convolution layer can affect the performance of CNN, we will used the AlexNet[9] as the comparison to the previous simple CNN model, however, we will do some modification for this network. Figure 5 shows the architecture of the AlexNet.

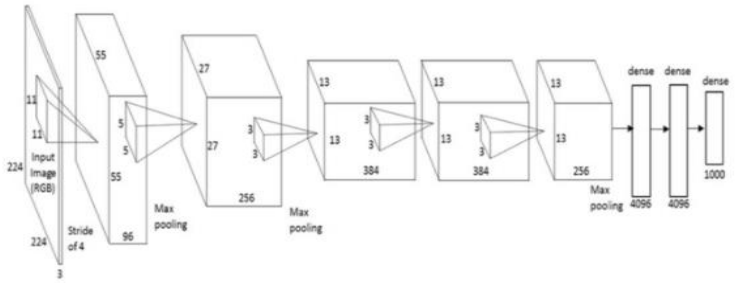

Fig. 5 AlexNet architecture[9]

AlexNet is a well-designed CNN architecture which achieved a decent result in training the ImageNet dataset. The AlexNet consists of eight layers, which are five convolutional layers and three fully connected layers. However, we will substitute the fully connected layers in the original model by using the FC net that is just introduced in figure 1.

2.5 Neuron pruning by distinctiveness

According to the research of T.D Gedeon and et al [10], pruning the hidden neurons will decrease the overfitting and increase the speed of computing. Pruning neurons by distinctiveness will be applied in the following testing.

Pruning network by distinctiveness includes four steps:

Firstly, apply different patterns on network structure, then train the neuron network with sufficient iterations, then normalize the weights value to clip the values between [-0.5, 0.5] (Eq. 5).

$$normalized = \frac{weight - \min(weight)}{\max(weight)} - 0.5$$

Secondly, extract the weight information of one of the hidden layers. All the extract weights will be stored in a matrix whose size of rows is the number of the output and size of column is the number of hidden neurons.

Thirdly, pick a pair of column vectors from the weight matrix and calculate the angle between those vectors. The angle between vectors can be calculated by equation (6), where A_i and B_i are different column vectors in weight matrix:

$$cos\theta = \frac{\sum_{i=1}^{n}(A_i * B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2 * \sum_{i=1}^{n}(B_i)^2}} = \frac{A^T B}{||A|| * ||B||} \tag{6}$$

Once we get the angle θ between those vectors, we could consider the pruning in two conditions. If the angle between the vectors is less than 15 degrees, then we remove one of the vectors, and the vector which is removed is added to the vector of unit which remains. If the angle between the vectors is larger than 165 degrees, we will remove both. The pruning will be applied during the training process. Since direct remove the weights during training will make the training dimension incompatible, all the vectors which should be removed will be set to zero.

2.6 Evaluation metric

Since there are not only simple classification tasks assessed (colorID 12 classes and typeID 11 classes), but also sophisticated classification task with 1,362 classes (vehicleID). The simple accuracy evaluation would be too harsh toward the complex case. Therefore, we introduce the top-K accuracy measuring method for our evaluation metric, here we will use the top-1 and top-3 criteria. Top-1 is equivalent to a simple accuracy metric which only determines whether the highest prediction matches the ground truth, while top-3 accuracy indicates that any of the highest 3 predictions match the ground truth. The formular is shown in Eq. 7.

$$Accuracy = \frac{\sum_{i=1}^{K} \delta_{top-k}(x_i, t_i)}{K} \tag{7}$$

K indicates the number of tested samples, x_i is the predicted result and t_i is the target.

3 Result and Discussion

3.1 Implementation setting and testing devices

All the models are trained by using RTX3070 8G GPU. Due to the memory limit, the maximum acceptable batch size is 50, therefore we use this batch size during the training. Each model is trained through 40 epoch, and random shuffle was used in each training epoch. All models are trained with the same training schedulers, which are Adam with learning rate 0.0001. In addition, during each epoch, the validation set is added after each training to evaluate the learning performance of current model (whether it is overfitting or not). The loss and accuracy data that will be displayed are the averaging values from each epoch. In another words, we will take the average value of all the loss/top-k accuracy we got to represent the current epoch loss/top-k accuracy. The loss will be the average loss of the three classes. Then we will compare their best results and converging speed and benchmark those models on the testing set and get the conclusion.

3.2 Loss evaluation

Followed by the settings discussed in section 3.1, we first compare the performance among the simple CNN, the FC net and AlexNet. Figure 6 shows the plots of training and validation loss of the FC net compares to the simple CNN and AlexNet. We can see that AlexNet and CNN have faster converging speed and less loss values than the FC net within the 40 epochs of training, while AlexNet has the fastest converging speed and lowest loss values.

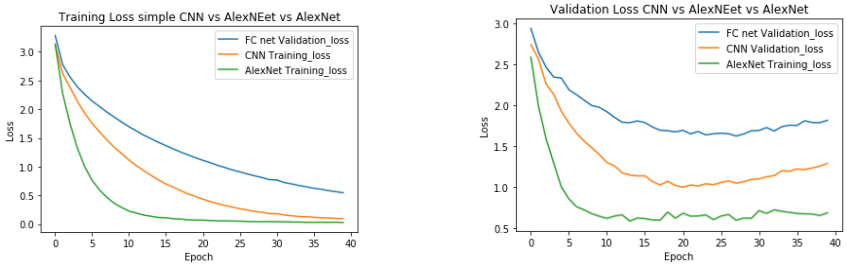

Fig. 6: Training loss and validation loss FC net vs CNN vs AlexNet

3.3 Accuracy evaluation

For the accuracy plots, the validation accuracy of all the models is shown in the figure 7. We can see that the majority of the experiment results indicates AlexNet has the highest accuracy and converging speed, simple CNN achieves the medium accuracy and converging speed while FC net has the lowest accuracy. However, the top-3 typeID is an exception, which shows that the simple CNN has the highest accuracy while AlexNet is medium.

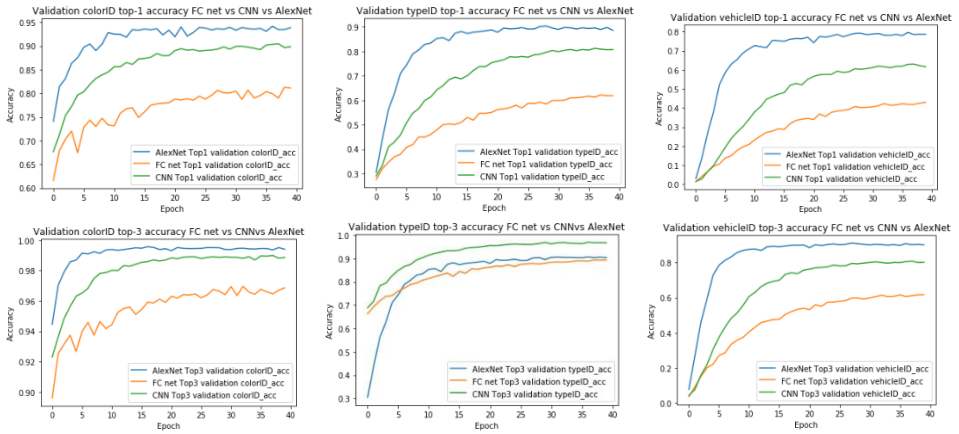

Fig. 7 Validation accuracy FC net vs CNN

3.4 Test results

Table 1 shows the results of test accuracy of the FC net, simple CNN and AlexNet. The comparison shows that AlexNet achieves the highest accuracy in all the testing classes and top-k criteria. Furthermore, for the vehicleID accuracy, which is a sophisticated classification task, its difference across models is much larger than other classes, especially for the FC net (19.02% less than CNN and 28.13% less than AlexNet). Even though the TypeID accuracy difference between the FC net and CNN is large under top-1 criteria (19.08% less than CNN), the difference is significantly reduced. The accuracy difference between different models is reduced by using the top-3 evaluation criteria, however FC net still has much lower accuracy of FC net than other two models. In addition, the difference of the vehicleID classification accuracy between the simple CNN and AlexNet is also larger than the difference between other ID accuracy.

Table. 1 Test accuracy FC net vs CNN vs AlexNet

Models	FC net	CNN	AlexNet
TypeID top-1 Acc %	62.14	81.22	88.59
ColorID top-1 Acc %	81.28	90.48	93.77
VehicleID top-1 Acc %	42.94	62.98	78.12
TypeID top-3 Acc %	89.38	96.89	98.44
ColorID top-3 Acc %	96.96	98.99	99.39
VehicleID top-3 Acc %	61.79	80.81	89.92

3.5 Models overfitting evaluation

To inspect the overfitting during training for each model, their plots of training loss against validation loss will be illustrated. Figure 8 shows the loss plots for each model, and table 2 shows the mean absolute difference between the total train loss and total validation loss.

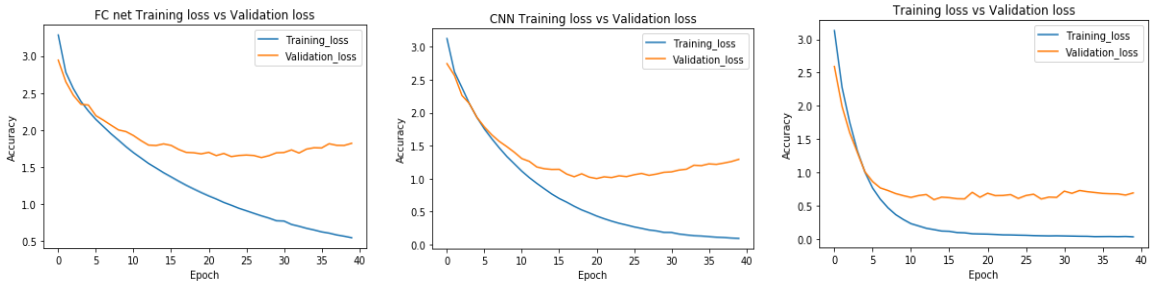

Fig. 8 Training loss against validation loss for each pattern

Table. 2 mean absolute value of the difference between the sum of training loss and validation loss for each model

FC net	CNN	AlexNet
0.5558	0.5503	0.4418

From the table 2, FC net and CNN has close value of the difference between training and validation loss, while AlexNet has the lowest difference. The mean loss between FC net and CNN is around 0.005, however the difference is around 0.11 between CNN or FC net and AlexNet. And figure 7 shows AlexNet has the highest similarity between the pattern of training loss and validation loss.

3.6 Neuron pruning

The figure 9 shows the validation loss results of each model.

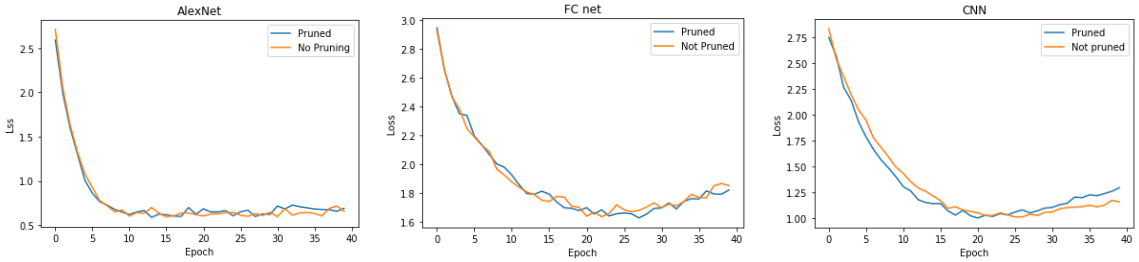

Fig. 9 Validation loss pruned vs not pruned

Although, the pruning could make the validation results different, it does not contribute much on improving the performance of the training, like reducing the overfitting. Similarly, the accuracy of all the tags with pruning do not have much variation compare to the results without pruning.

3.7 Discussion

According to the experiment results from the previous sections, we proved that CNN could achieve outstanding performance than the normal fully connected in image classification task. The loss and accuracy converging speed will be faster if CNN based method is used during training. In another aspect, when doing a complex classification task like the vehicleID classification, CNN could learn the complex features in an image much better the fully connected network. These results may be since CNN has a variety of dimension reduction procedures inside significantly reduces the complexity of the image input, therefore reduce the workload and overfitting in the FC layers. Besides, features of the image will be learned in a more systematical way through the convolutional kernels.

Furthermore, the advanced fitting performance could be achieved by stacking more convolutional layers in a CNN. The comparison between the simple CNN and AlexNet has proved this assumption. From the section 3.4 we could find that the complex feature classification will be learned better in a deeper convolution layer. From the section 3.5 we could find the deeper convolution layer would also make the training become more robust (reduce the overfitting and stabilize the loss pattern).

For the pruning results, there is no significant difference brought by this method in terms of less overfitting or faster converging speed. This maybe because the pruning angles used does not fit this model or the pruning layers are too less to provide large contribution.

4 Conclusion and Future work

This paper investigates how the CNN can achieve a better fitting in image classification tasks than a fully connected network and the deeper convolution layers could obtain advanced performance than a shallow convolution neural network. The evidence is collected by testing the different multi-class classification in VehicleX dataset. It can be concluded that CNN can achieve a better fitting in image classification tasks than a fully connected network in terms of converging speed, complex features fitting, and deeper CNN can achieve better performance than the shallow CNN in terms of same phases. For the pruning method, it does not bring much improvement for the training process.

For the future work, we will examine whether by keep adding the convolutional layers for a deep CNN will result better performance. We will present the experiment by selecting more CNN based models such as VGG net and deep ResNet. We will evaluate their performance by looking at the testing accuracy, time and special complexity and the overfitting pattern.

5 Reference

1. Kaewkamnerd, S., Pongthornseri, R., Chinrungrueng, J., Silawan, T.: Automatic vehicle classification using wireless magnetic sensor. Proc. 5th IEEE Int. Work. Intell. Data Acquis. Adv. Comput. Syst. Technol. Appl. IDAACS'2009. 420–424 (2009). https://doi.org/10.1109/IDAACS.2009.5342949.
2. Liu, W., Zhao, X., Xiao, J., Wu, Y.: Automatic vehicle classification instrument based on multiple sensor information fusion. Proc. - 3rd Int. Conf. Inf. Technol. Appl. ICITA 2005. I, 379–382 (2005). https://doi.org/10.1109/icita.2005.82.
3. Cheung, S.Y., Coleri, S., Dundar, B., Ganesh, S., Tan, C.W., Varaiya, P.: Traffic measurement and vehicle classification with single magnetic sensor. Transp. Res. Rec. 173–181 (2005). https://doi.org/10.3141/1917-19.
4. Kasar, M.M., Bhattacharyya, D., Kim, T.H.: Face recognition using neural network: A review. Int. J. Secur. its Appl. 10, 81–100 (2016). https://doi.org/10.14257/ijsia.2016.10.3.08.
5. Redmon, J., Farhadi, A.: YOLO v.3. Tech Rep. 1–6 (2018).
6. Singh, S.P., Kumar, A., Darbari, H., Singh, L., Rastogi, A., Jain, S.: Machine translation using deep learning: An overview. 2017 Int. Conf. Comput. Commun. Electron. COMPTELIX 2017. 162–167 (2017). https://doi.org/10.1109/COMPTELIX.2017.8003957.
7. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 39, 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031.
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-Decem, 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90.
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM. 60, 84–90 (2017). https://doi.org/10.1145/3065386.
10. Gedeon, T.D., Harris, D.: Network Reduction Techniques. 1–7 (1988).
11. Yao, Y., Zheng, L., Yang, X., Naphade, M., Gedeon, T.: Simulating Content Consistent Vehicle Datasets with Attribute Descent. In: ECCV (2020).