CSC8635 Report

December 31, 2021

1 Introduction

[ ]:
```
from IPython.display import Image
```

Drivers' states while driving is important in keeping driving save. Drinking, Talking while driving are critical factors that will impact the safety while driving. Therefore designing a program that can automatically detect drivers' status is important. State Farm Distracted Driver Detection Dataset contains nine dirver status: safe driving, texting, talking on the phone, operating the radio, reaching behind, makeup and talking to passenger.

In this project, A deep neural network model, MobileNet, is used to classify images. In addition, by comparing the effects of super-parameter adjustment on experimental results, this paper explores how super-parameters influence training process and training accuracy.

2 Methods

2.1 Deep Leaning method: MobileNet

There are many deep learning methods that can be used to classify images. Classic convolutional neural network based image classification techniques include Lenet, VGG, etc. In general, image classification technology based on deep learning firstly uses multi-layer convolution module to extract features from images, and then uses fully connected layer to transform the learned features into a one-dimensional vector. And then categorize and predict. The number of neurons in the output layer is the number of final classifications.

In this project, MobileNet[1] is a light weight deep neural network model that uses depth-wise separable convolutions rather than traditional convolutions. With deep-wise separable convolutions, MobileNet's parameter number is significantly less than that of , and the network is also faster to train. Accordingly, MobileNet is suitable to be deployed on mobile devices.

Standard convolutional layer is composed of a 3x3 convolutional layer, a batch normalization layer and a ReLU activation layer. However, in depth-wise separable convolutional layer, the 3x3 convolutional layer is replaced by 3x3 Depthwise convolutional layer. And subsequently, a 1x1 convolutional layer with a batch normalization layer and a ReLU layer is followed. The following figure shows the comparison between the two structures.

MobileNet uses a combination of deep-wise convolutional layers and standard convolutional layers to perform image feature extraction task. As compared to Conv MobileNet (MobileNet that built with full convolutions), MobileNet with depth-wise separable convoutional layers has slightly poor

performance (71.7% and 70.6% on ImageNet). But the latter has significantly smaller number of parameters (29.3 millions versus 4.2 millions).

2.2 Experiment Process

There are three files in this project: *load_data.py*, *main.py*, *VGG.py*. The first is used to read raw image data to tensor. Notably, when it comes to training data, *RandomHorizontalFlip* method is used to flip training images horizontally. *load_data.py* also seprates training data and validation data, using traindata_size and valdata_size to adjust. The design of *load_data.py* also uses special methods to make the code more efficient. *main.py* includes parameter setting function, training and validation program. Validation set is implemented immediately after training to obtain the performance of the current model parameters. After finishing each epoch, the model is saved as *model.pt*. *VGG.py* includes the VGG16 and the VGG19 neural network model that I built by myself. Unfortunately, this porject did not use this model due to its effeciency. The neural network is trained on an Nvidia RTX 3060ti with CUDA enabled.

2.3 Parameters comparision

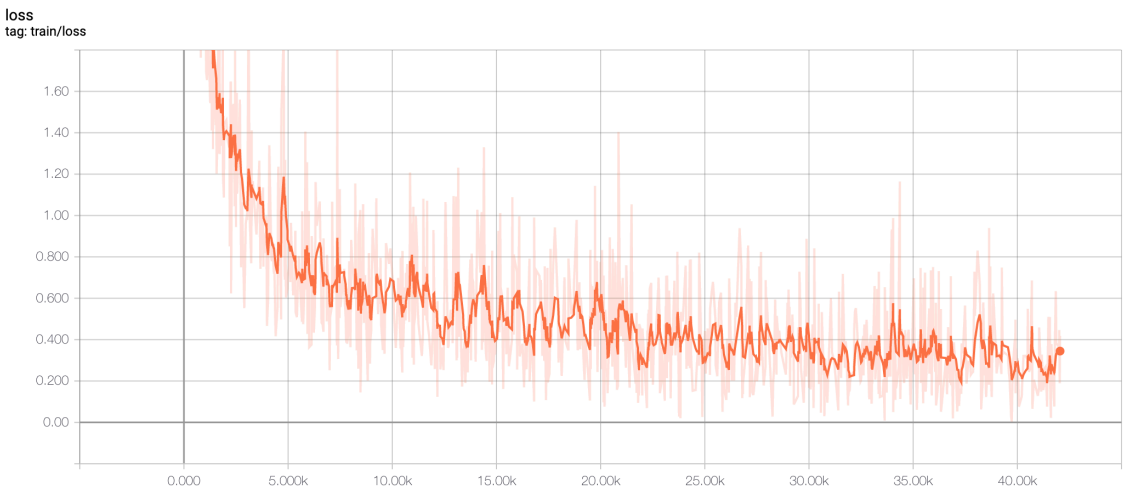
This section reports experiment result. There are six expirements in total, and in each of them, one parameter is modified. The first experiment group is

2.3.1 Standard Group

This experiment group set epoch number as **50**, learning rate as **0.0001**, batch size as **16** and dataset parameters as **0.6 and 0.95**. Following is the training process:

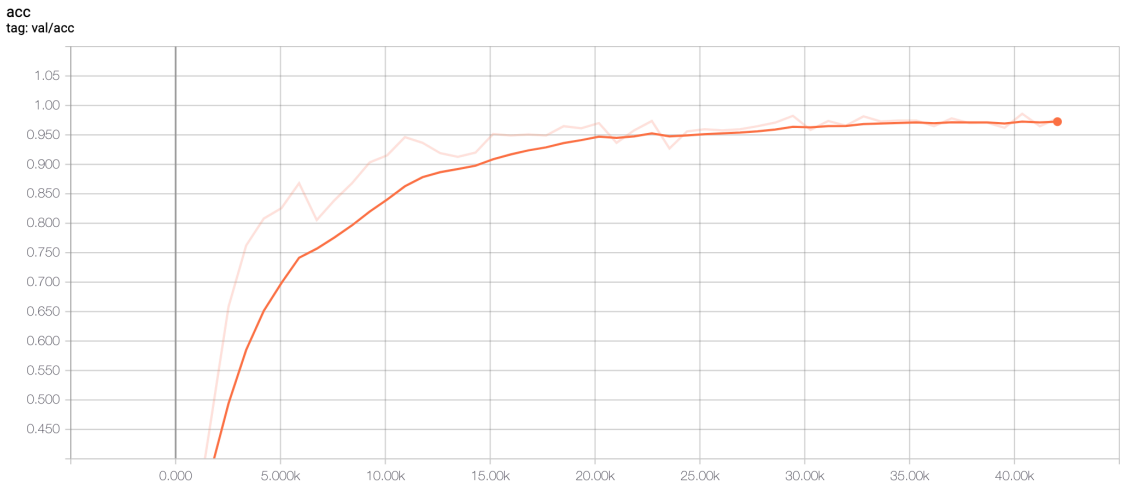
[ ]:
```
Image("./pics/SG_1.png")
```

[ ]:



[ ]:
```
Image("./pics/SG_2.png")
```

[ ]:

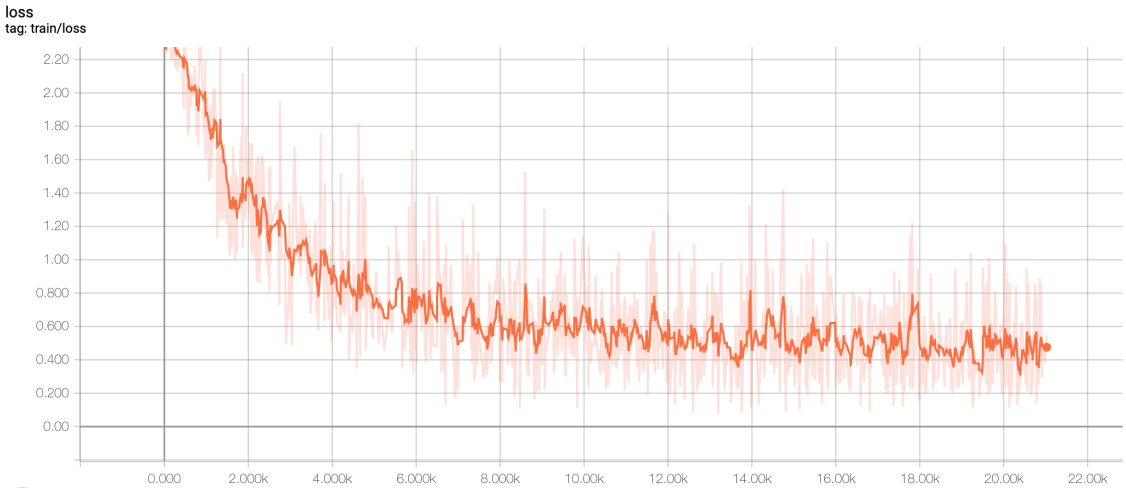


2.3.2 Different Epoch

This experiment group set epoch number as 25, learning rate as 0.0001, batch size as 16 and dataset parameters as 0.6 and 0.95. Following is the training process:

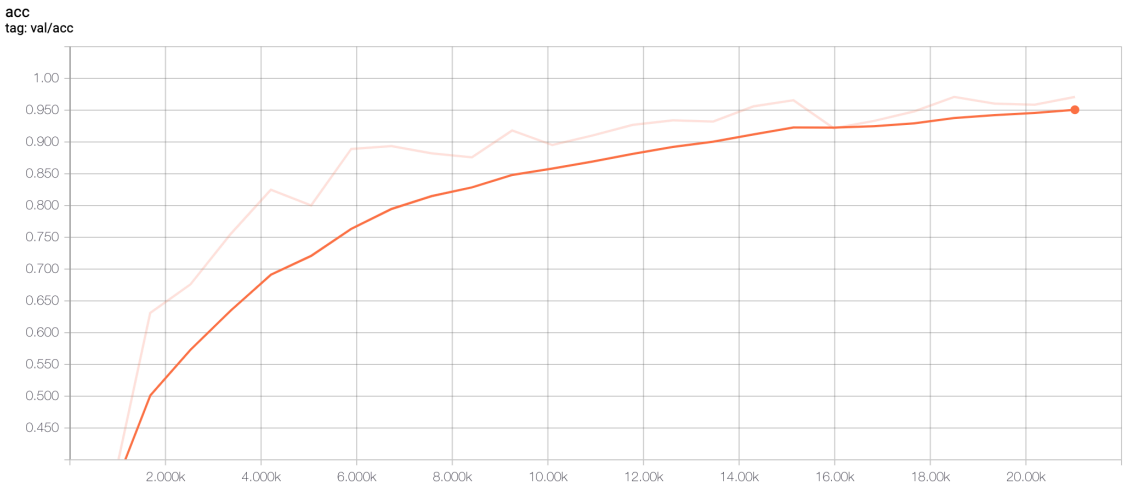
[ ]:
```
Image("./pics/Epoch_1.png")
```

[ ]:



[ ]:
```
Image("./pics/Epoch_2.png")
```

[ ]:

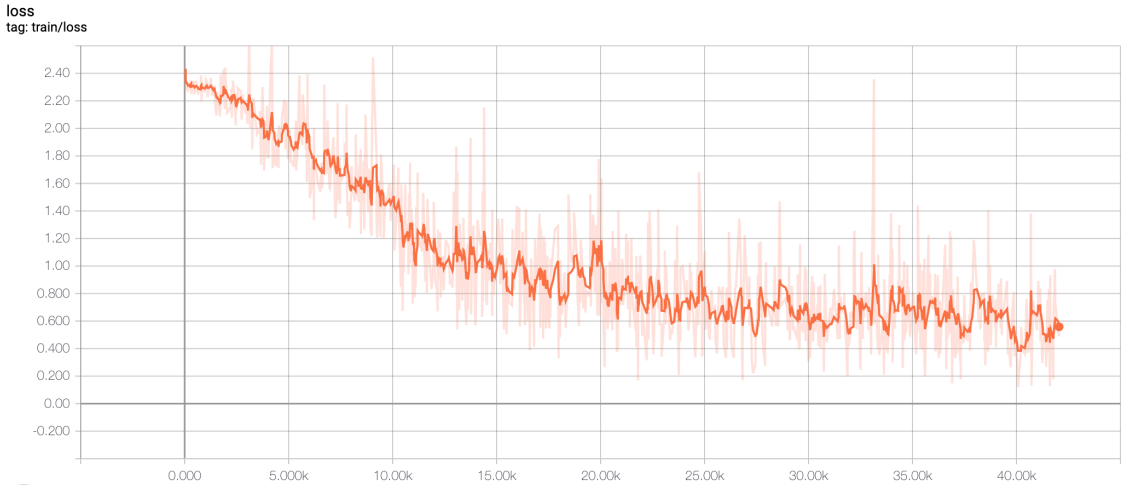


2.3.3 Different Learning Rate

This experiment group set epoch number as 50, learning rate as 0.001, batch size as 16 and dataset parameters as 0.6 and 0.95. Following is the training process:

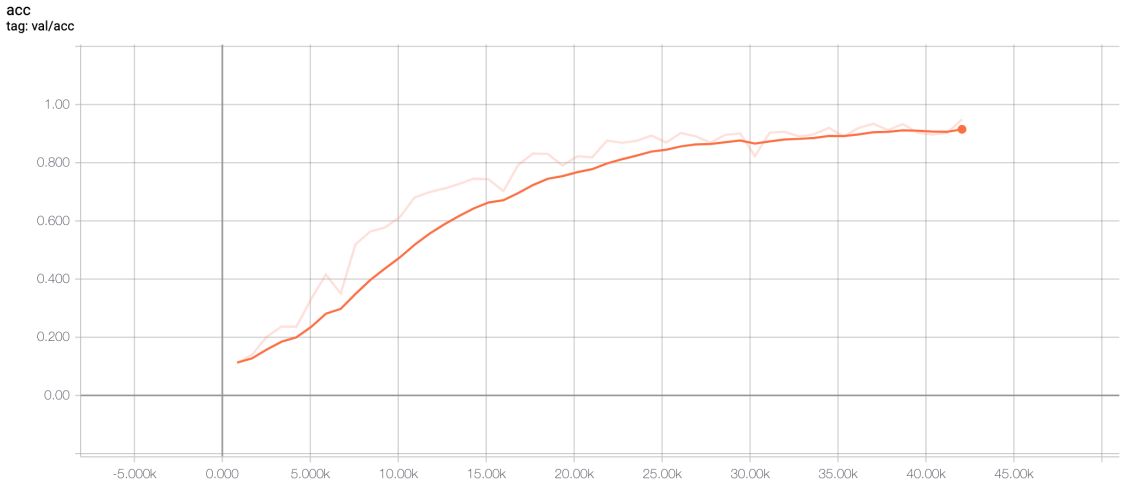
[ ]:
```
Image("./pics/LR_1.png")
```

[ ]:



[ ]:
```
Image("./pics/LR_2.png")
```

[ ]:

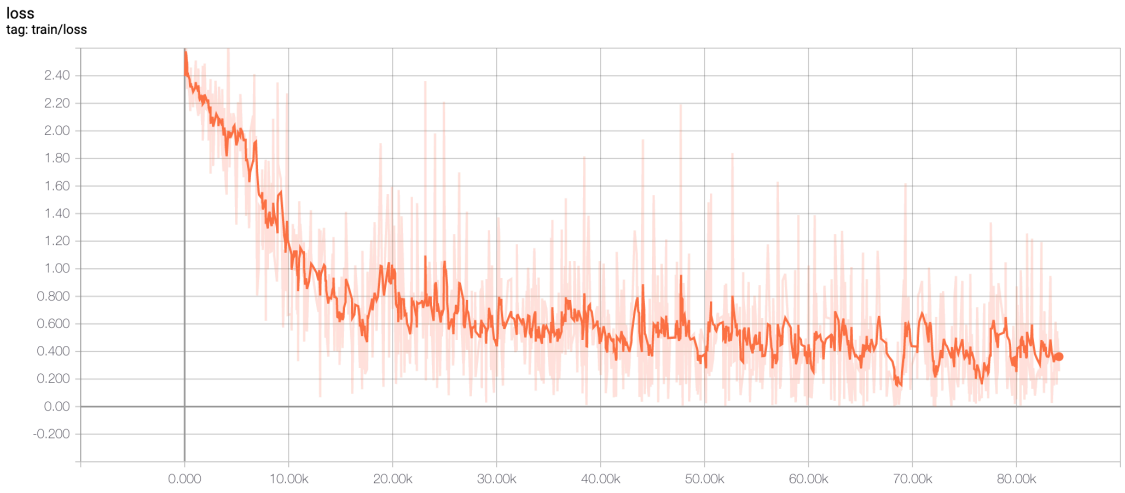


2.3.4 Different Batch Size

This experiment group set epoch number as 50, learning rate as 0.0001, batch size as 8 and dataset parameters as 0.6 and 0.95. Following is the training process:

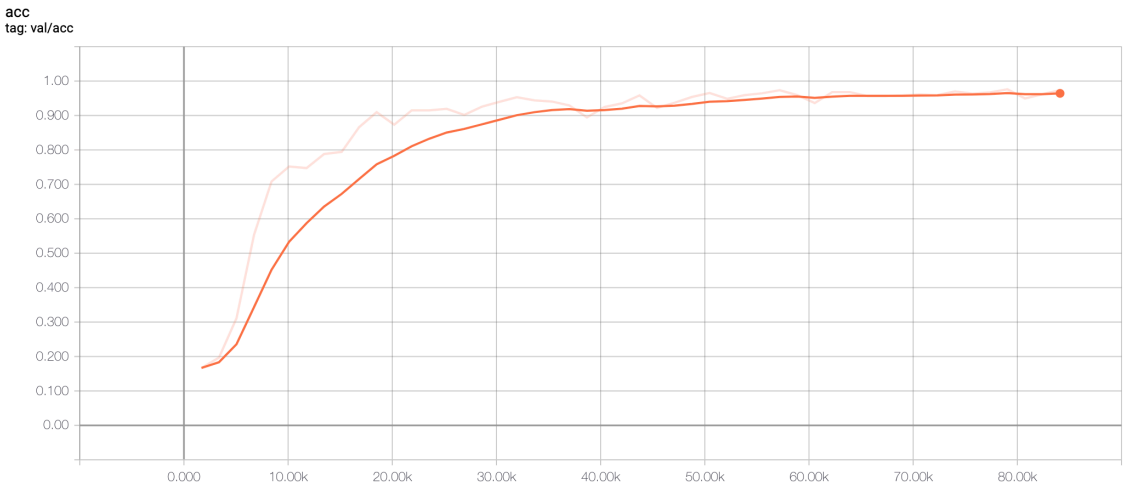
[ ]:
```
Image("./pics/BS_1.png")
```

[ ]:



[ ]:
```
Image("./pics/BS_2.png")
```

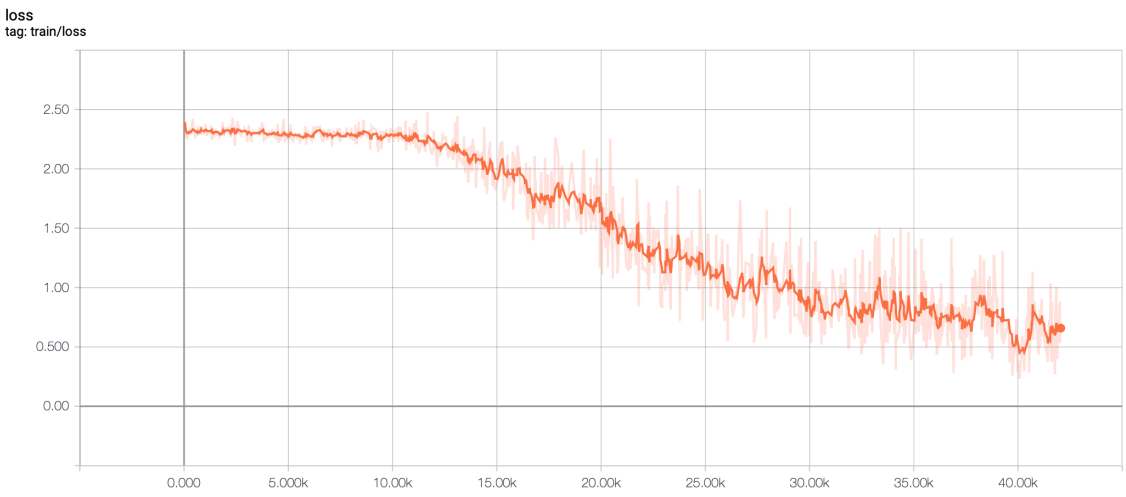
[ ]:



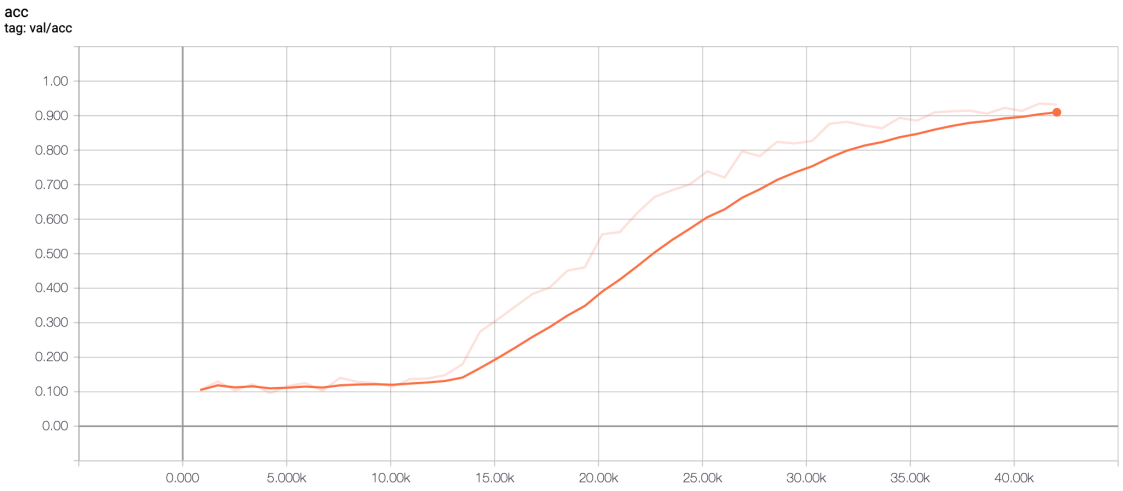
2.3.5 SGD over Adam

This experiment group set epoch number as 50, learning rate as 0.0001, batch size as 16 and dataset parameters as 0.6 and 0.95. However, the optimizer uses SGD rather than Adam. Following is the training process:

[ ]:
```
Image("./pics/SGD_1.png")
```
[ ]:



[ ]:
```
Image("./pics/SGD_2.png")
```
[ ]:



2.3.6 Different Training Batch Size

This experiment group set epoch number as 50, learning rate as 0.0001, batch size as 16 and dataset parameters as 0.6 and 0.95. Following is the training process:

2.4 Analysis

3 Conclusion

4 Discussion

5 Reference

[1] MobileNet