Machine Learning Project Report

C-VGG-19

Chencong Du
Zhuoqi Zhang
Ao Wang

1. Introduction

Image recognition is a popular application of machine learning. Among various approaches, the Convolutional Neural Network (CNN) has successfully been applied into visual image analyzes. In ImageNet ILSVRC, CNNs have shown their reliable performances in recent years compared with traditional computer vision methods. In the project, CNN is chosen to be the core method to achieve our objective.

A typical CNN is constructed by convolutional layers and fully connected layers. Due to the problem of time consumption and difficulties of parameter tunings to train a CNN from very beginning, we tried to start from a pretrained CNN and fit a new model with our new dataset. That is using

1.1 Expectation

With a set of car images and correctly classified labels indicate a specific model, type name and generation year of a car, we want to train a CNN to classify the new cars in images. We also tried to fit a SVM and Naive Bayes model using features extracted from pretrained CNNs to see their performances.

1.2 Technical Method

In this project, we chose the Matlab as the development environment. Because the pretrained Alexnet model (https://cn.mathworks.com/matlabcentral/fileexchange/59133-neural-network-toolbox-tm--model-for-Alexnet-network) and VGG-19 model (https://cn.mathworks.com/matlabcentral/fileexchange/61734-neural-network-toolbox-model-for-VGG-19-network) have existed. We could use them directly. Using the pretrained model will save a lot of time, then we could concentrate on how to transfer those models to recognize the cars in images.

To make the process of training more efficient, we need to train the model on the GPU. Due to GPU in our PC cannot meet the stand to finish training. So, we chose the Amazon EC2. The specific instance type we used is P2.xlarge.

OS platform: Windows.

CPU: Intel Xeon E5-2686 v4

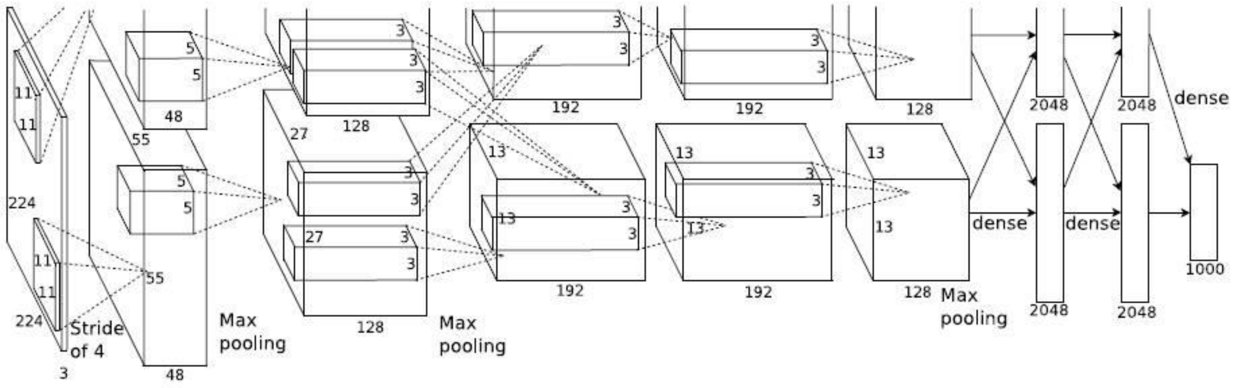
GPU: NVIDIA K80 GPUs, each with 2,496 parallel processing cores

Mem: 61GiB

GPU Mem: 12 GiB

2. Model & Dataset

a. Alexnet



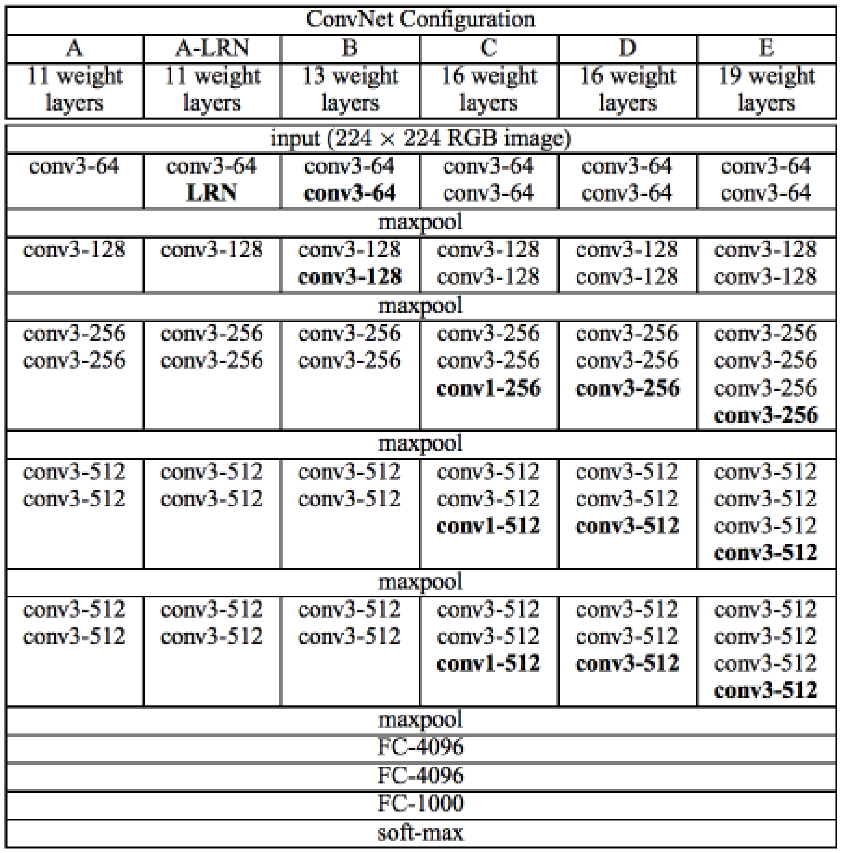
In ILSVRC2012, AlexNet was proposed. It's the winning model of ILSVRC2012 classification task and it achieved a large accuracy margin compared with the non-DNN methods. It contains 5 convolutional layers and 3 fully-connected layers. During training, some randomness is introduced in the data augmentation process and dropout layer.

b. VGG-19

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper, Very Deep Convolutional Networks for Large Scale Image Recognition.

This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier (above).

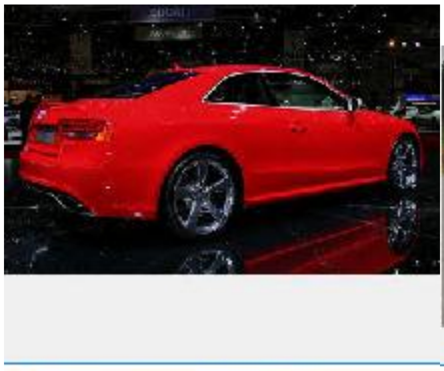
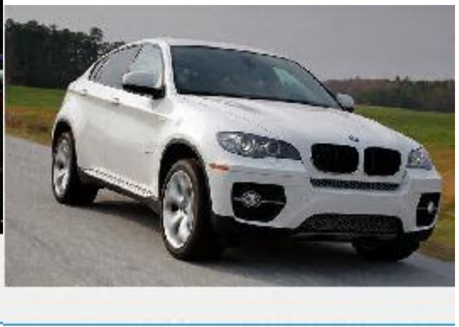
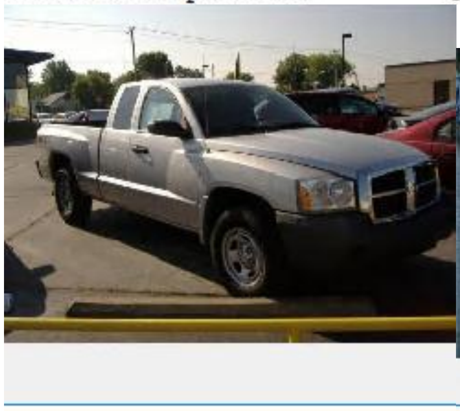
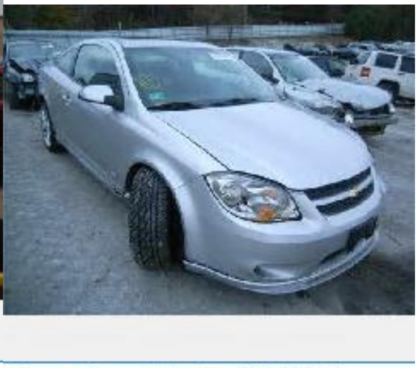
The “19” stand for the number of weight layers in the network (columns E in Figure below):



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

c. Dataset

Because the expectation, the suitable dataset is Cars Dataset from Stanford. This dataset contains 196 classes of cars. The number of training images is 8144. From the picture below, we can get that the labels are typically at the level of Make, Model, Year, e.g. Audi S5 Coupe 2012 or BMW X6 SUV 2012.


```
Audi S5 Coupe 2012.
```

```
car is -------- BMW X6 SUV 2012.
```

```
Dodge Dakota Club Cab 2007.
```

```
Chevrolet Cobalt SS 2010.
```

This dataset also gives us the boundary of the car in every image. In other word, we could use those data to outline the car in every picture.

d. Cars Dataset vs ImageNet

In our project, our aim is to identify the model and type of the car in an image. ImageNet is to identify the different classes in one image, e.g. plants, animals, person.

Our project subdivides the classes of car. because we did not only identify there is a car in the image, also we would give the model and other information of this car.

3. Implementation

3.1 Why Transfer Learning

Basically, transfer learning is to utilize the commonality between different knowledges, to acquire the knowledge we don't know based on the knowledge we have already learnt. For instance, it won't take too much time for a Java engineer to learn C++ since the prerequisites of these two languages share a lot of similarities.

Nowadays, transfer learning has been widely applied into many problems where there are some connections between one or more existed models and new problems. In this project, the dataset consists images, the model we want to train is CNN classification model. Obviously, the characters of this problem share many common points with ImageNet ILSVRC, so naturally, we think transfer learning based on champions of ImageNet ILSVRC will provide a good performance.

In particular, there are some typical methods to do transfer learning (compared with dataset of pretrained model):

(1) dataset is small, and has high similarity.

In this case, we only have to change the output number of last layer of pretrained model, to fit the training data and labels we have now.

(2) dataset is small, and has low similarity.

In this case, we can only train the last one or several layers in the model, and fix the weights of other layers. Since the similarity is low, so it will take skilled parameter tuning to get a well-trained model.

(3) dataset is big, and has low similarity.

In this case, a big dataset can be both an advantage and disadvantage. Since the training procedure will be effective with a big dataset, while the low similarity will impact the pretrained model. So, we'd better reset all weights and retrain the whole model with new dataset.

(4) dataset is big, and has high similarity.

This is the ideal case, we can just keep the structure and initial weights of model and retrain the pretrained model with new dataset.

3.2 Transfer Learning on Alexnet

Now back to our dataset, we can see that the champion models of ImageNet ILSVRC is able to effectively classify the objects according to categories while our images are all cars, we need to get features from car images to predict the model, type, and generation year. So, the similarity between the datasets of pretrained model and our model is low. And compared to ImageNet ILSVRC, our dataset is relatively small. Like we mentioned above, we are in case 2 now. First, we focus on Alexnet, which has 8 layers in total, 5 of them are convolutional layers, followed by fully connected layers. And here we come up with several ideas of how to do the transfer learning based on Alexnet with our dataset:

(1) Retrain the last one fully connected layer

(2) Retrain the last two fully connected layers

(3) Add another fully connected layer

(4) Extract features from pretrained Alexnet and fit a SVM

(5) Extract features from pretrained Alexnet and fit a naive bayes model

3.3 Challenge & Optimized Solutions

a. Training Process

We train the model on our PC first, but the elapsed time is too long because the capacity of GPU on the computer is poor to finish this work. So we use Amazon EC2 to set up a instance which have high computational capabilities. With the high capacity GPU, the training process runs well, although the training process still cost a lot times.

b. Deeper Neural Network

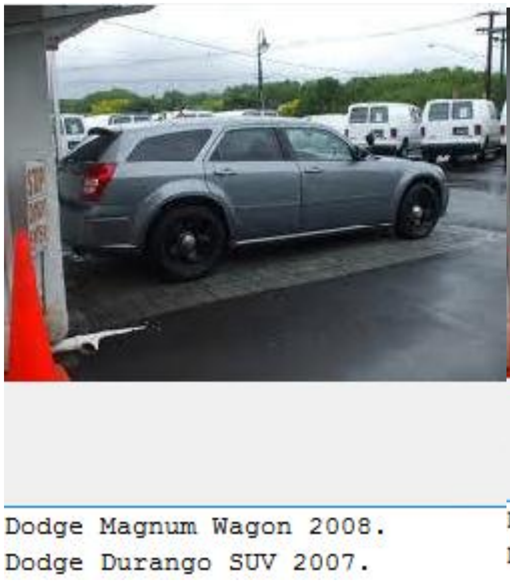
After finishing training Alexnet, we find that the test accuracy is poor, no matter we retrained the last layer, last two layers or add one more layer. So, we consider the reason of this result is because the Alexnet is still not complex. Then we use another CNN model, VGG-19, which is more complex than Alexnet. The result of accuracy with this model is very good.

c. Preprocess Dataset

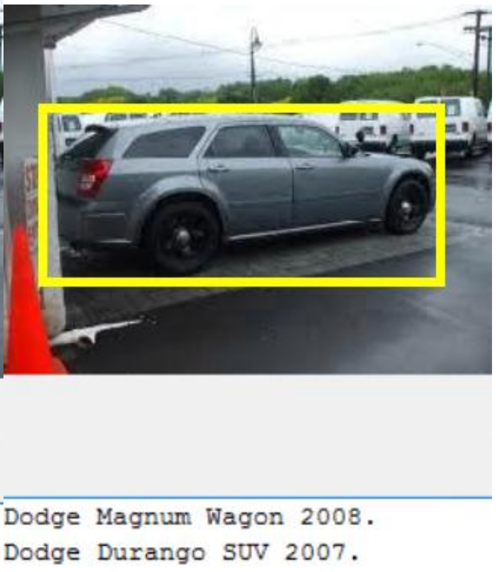
After solving the problems above, we use the Cars dataset to train out model. But the result of accuracy is not very acceptable. We pick some images that our model make a wrong prediction to do a little research.

We find those images which are hard to be identified has a lot of irrelevant items. Like the image below, there are a lot of irrelevant cars beside the target car. Furthermore, we can only get the side of the car. So those factors would affect to the accuracy.

What we have done is that we outline the car in the image according to the dataset (the result shown in the below figure). This move will help our model to identify the target car. No matter there are some irrelevant things in one image.



```
Dodge Magnum Wagon 2008.
Dodge Durango SUV 2007.
```



```
Dodge Magnum Wagon 2008.
Dodge Durango SUV 2007.
```

3.4 Transfer Learning on VGG-19 with optimized dataset

Then we take VGG-19 as model we are going to transfer from, and put all our efforts on adding another fully connected layer in VGG-19, since this method provides the best result. And here are our new ideas:

(1) Add another fully connected layer in Alexnet with optimized dataset.

(2) Add another fully connected layer in VGG-19 with original dataset.

(3) Add another fully connected layer in VGG-19 with optimized dataset.

You may wondering why not just try the 3th method, since the time cost of training VGG-19 is ridiculously long compared to Alexnet. So, we just confirm our optimization method step by step, to see how the VGG-19 and modified dataset performs. Then we combine these two methods and get the best result eventually.

4. Result

We have done five training with the Alexnet model.

The table below shows the results of every test. We can see that the testing accuracy is always poor, no greater than 40%. As for the SVM and Naive Bayes method, the accuracy of those method is not worth to comparison.

	Last 2 'FC'	Last 1 'FC'	Add 1 'FC'	SVM	Naive Bayes
Training Accuracy	79.69%	79.30%	77.24%	---	---
Test Accuracy	33.92%	38.30%	40.37%	32.95%	15.51%
Elapsed Time	913 s	906 s	933 s	---	---

Optimized solution

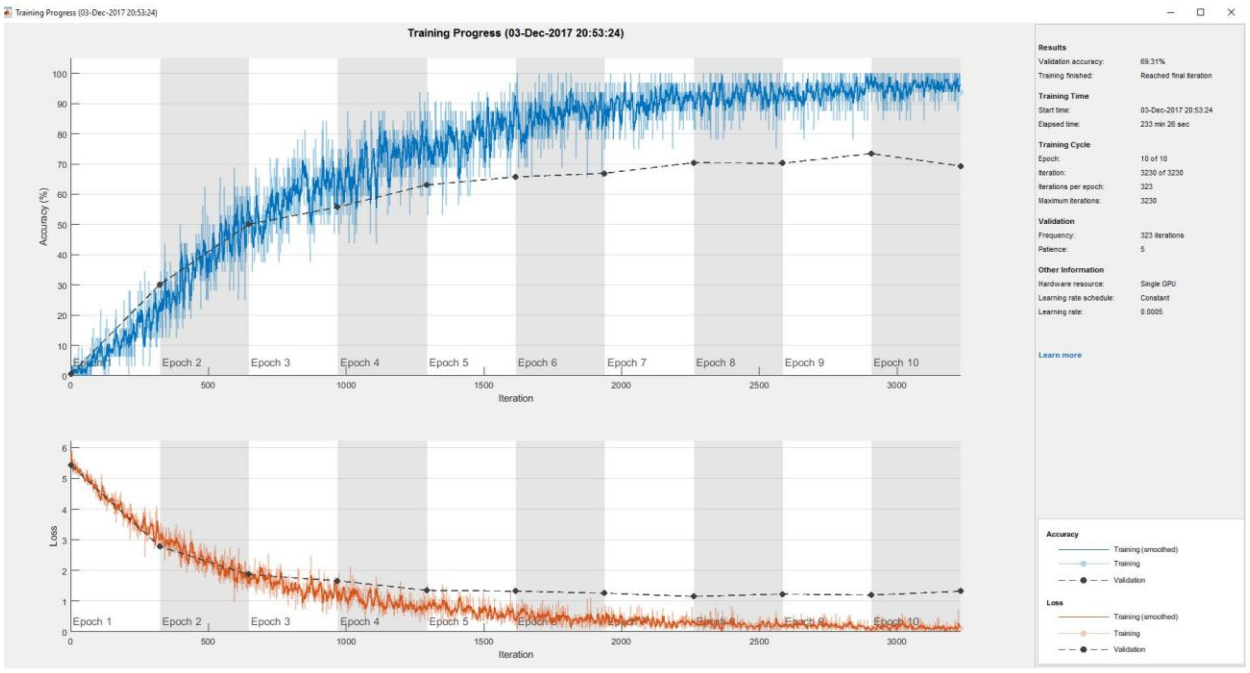
	VGG-19 with Original Dataset	Alexnet with Optimized Dataset	VGG-19 with Optimized Dataset
Training Accuracy	92.19%	93.759%	95.94%
Test Accuracy	68.192%	64.359%	79.227%
Elapsed Time	233 min 26 sec	10min 53 s	210 min 43 sec

This table show the result of the optimized result.

After we apply the optimization solution to our model. We can see the test accuracy is higher than the test before.

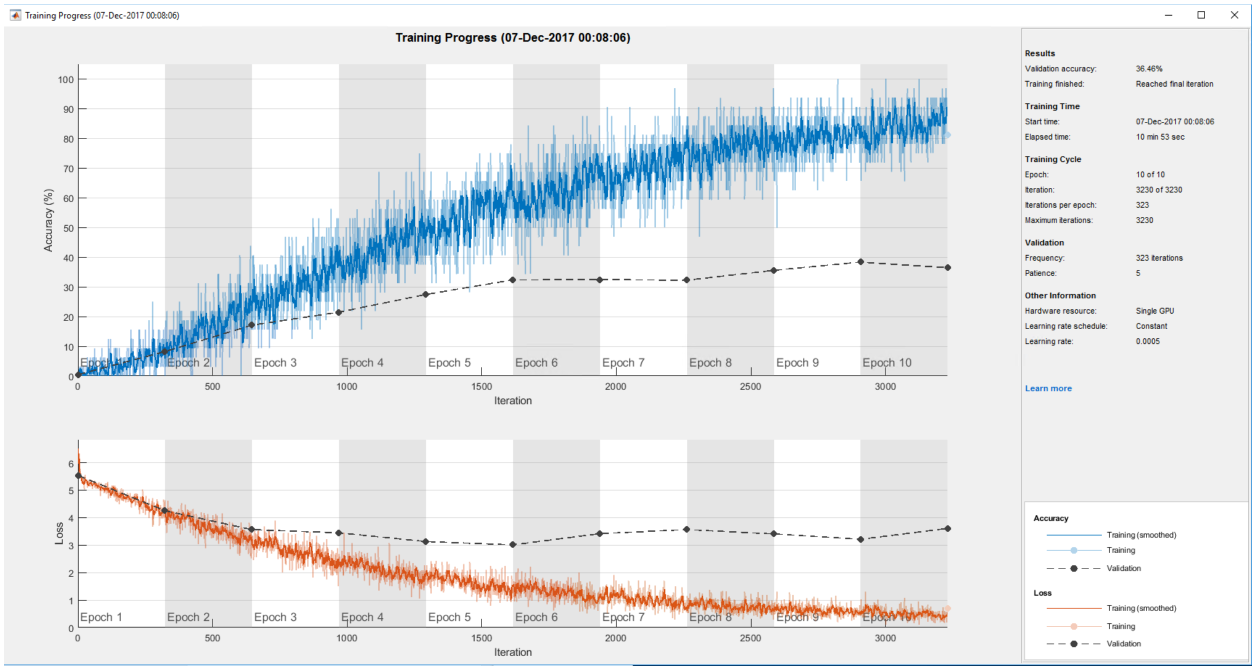
Training progress of three optimized solutions.

1. Deeper Neural Network with original dataset



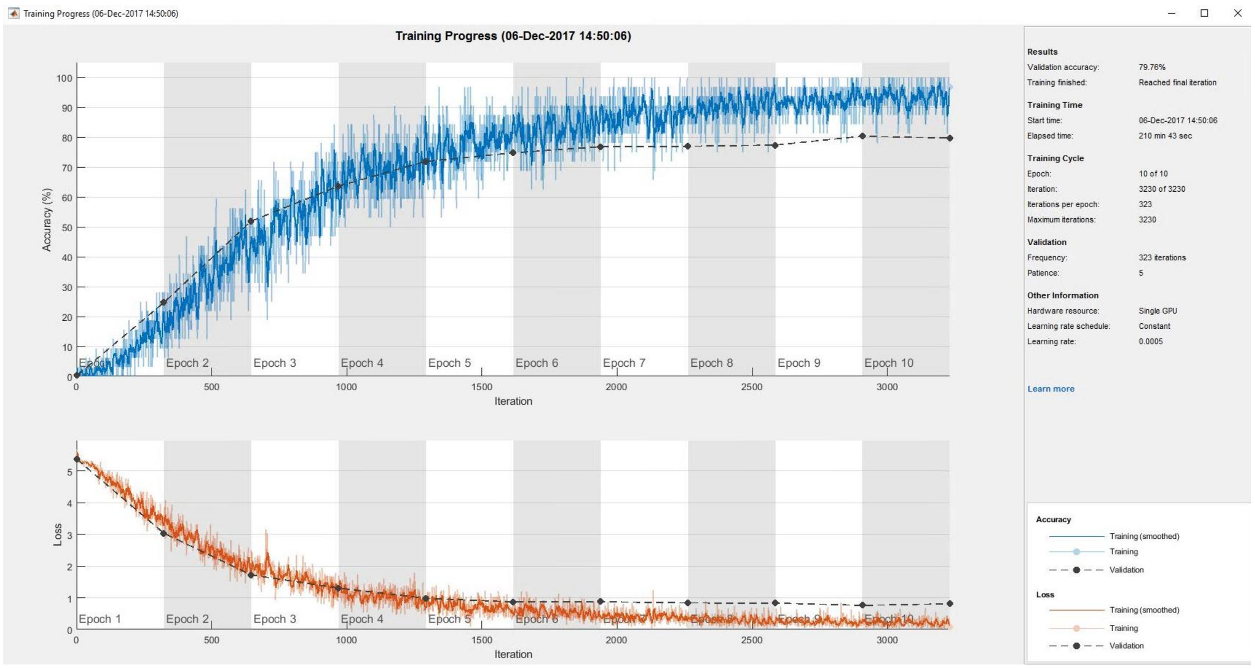
Results	
Validation accuracy:	69.31%
Training finished:	Reached final iteration
Training Time	
Start time:	03-Dec-2017 20:53:24
Elapsed time:	233 min 26 sec
Training Cycle	
Epoch:	10 of 10
Iteration:	3230 of 3230
Iterations per epoch:	323
Maximum iterations:	3230
Validation	
Frequency:	323 iterations
Patience:	5
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0005

2. Alexnet with optimized dataset



Results	
Validation accuracy:	36.46%
Training finished:	Reached final iteration
Training Time	
Start time:	07-Dec-2017 00:08:06
Elapsed time:	10 min 53 sec
Training Cycle	
Epoch:	10 of 10
Iteration:	3230 of 3230
Iterations per epoch:	323
Maximum iterations:	3230
Validation	
Frequency:	323 iterations
Patience:	5
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0005

3. Deeper Neural Network with optimized dataset



Results

Validation accuracy:	79.76%
Training finished:	Reached final iteration

Training Time

Start time:	06-Dec-2017 14:50:06
Elapsed time:	210 min 43 sec

Training Cycle

Epoch:	10 of 10
Iteration:	3230 of 3230
Iterations per epoch:	323
Maximum iterations:	3230

Validation

Frequency:	323 iterations
Patience:	5

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0005

Learn more

5. Project Overview

In this Project, we have implemented the cars information classification. The model could identify the cars model and generation, also the accuracy could reach to almost 80%.

The method we choose is transfer learning. We retrain the specific layers in existed model (Alexnet and VGG-19). From the training process, we find some weakness in the model based on Alexnet. We come up with three optimization solutions to solve the shortage. In the end, the accuracy of the final model we got is acceptable.

In the future, we consider that we could use RNN to outline the cars in image automatically. In our project, we outline the car in images based on the dataset. And after the testing, outlining the car in image could help a lot to classification model.

6. Reference

Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 554-561).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).