CMP 719 - Computer Vision Assignment 1

Aytekin YILDIZHAN

N18147923

PART – I

In this homework, our data is Stanford Dogs Dataset [1]. It consists of 120 classes of dogs. We use just 20 classes of them which are Chihuahua, Irish_terrier, Kerry_blue_terrier, Australian_terrier, Norfolk_terrier, Yorkshire_terrier, African_hunting_dog, Pug, Rottweiler, Redbone, English_foxhound, Shih-Tzu, Sussex_spaniel, German_shepherd, French_bulldog, malamute, Eskimo_dog, Siberian_husky, Chow and Dingo. Table 1 shows the number of dog species. We have 3406 dog images.

Table 1. Number of dogs

Label	Count
n02085620-Chihuahua	152
n02086240-Shih-Tzu	214
n02089973-English_foxhound	157
n02090379-redbone	148
n02093859-Kerry_blue_terrier	179
n02093991-Irish_terrier	169
n02094114-Norfolk_terrier	172
n02094433-Yorkshire_terrier	164
n02096294-Australian_terrier	196
n02102480-Sussex_spaniel	151
n02106550-Rottweiler	152
n02106662-German_shepherd	152
n02108915-French_bulldog	159
n02109961-Eskimo_dog	150
n02110063-malamute	178
n02110185-Siberian_husky	192
n02110958-pug	200
n02112137-chow	196

In Part I, when we examine the data, the sizes of the dog images are different from each other. Therefore, we re-sized the original image to the size of 256 x 256 x 3 since it is therefore necessary to convert all our images to same size to define architecture of our model.

Secondly, we split the data. For each classes, 60% of data is used to training, whereas the 20% is used for the testing and the remaining 20% for the validation randomly. During the process, we try different ratio because these ratio is problem-dependent.

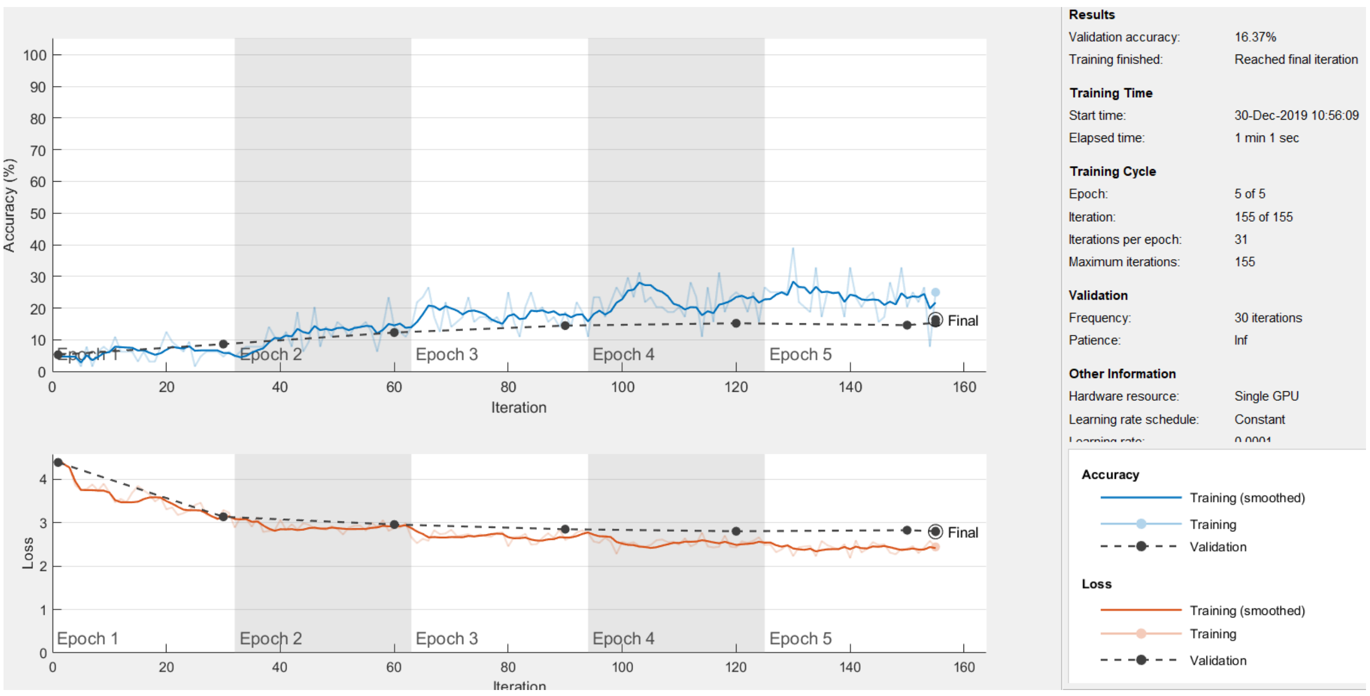
Finally, data augmentation can improve the models ability to generalize and correctly label images so we implement data augmentation method, such as resizing, rotation, shifting, adjusting brightness, shearing intensity, zooming and flipping.

PART – II

In Part II, we build a CNN that train a network from scratch. First architecture and hyperparameter are following. We'll initialize these filters and parameters randomly.

```
layers = [

    imageInputLayer([256 256 3])

    convolution2dLayer(3,8,'Padding','same')
    reluLayer

    convolution2dLayer(3,16,'Padding','same')
    reluLayer

    convolution2dLayer(3,32,'Padding','same')
    reluLayer

    fullyConnectedLayer(20)
    softmaxLayer
    classificationLayer];


options = trainingOptions('sgdm', ...
    'MiniBatchSize',64, ...
    'InitialLearnRate',0.0001, ...
    'MaxEpochs',5, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imageValidation2, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

In first experiment, validation accuracy is %10.53. Our loss is stable at 3 so we change our parameter. Firstly we add `batchNormalizationLayer` after each `convolution2dLayer`. When add this component, elapsed time is increasing because the layer dimensions and parameters are high. Then, validation accuracy is %12.72 but our loss is increasing, %6.5. We need to decrease the elapsed time so we add `maxPooling2dLayer` after each `reluLayer`. Then we train network. The validation accuracy is %16.37. Our loss is nearly stable at 3. The curves are shown in Figure 1.



Results
Validation accuracy: 16.37%
Training finished: Reached final iteration

Training Time
Start time: 30-Dec-2019 10:56:09
Elapsed time: 1 min 1 sec

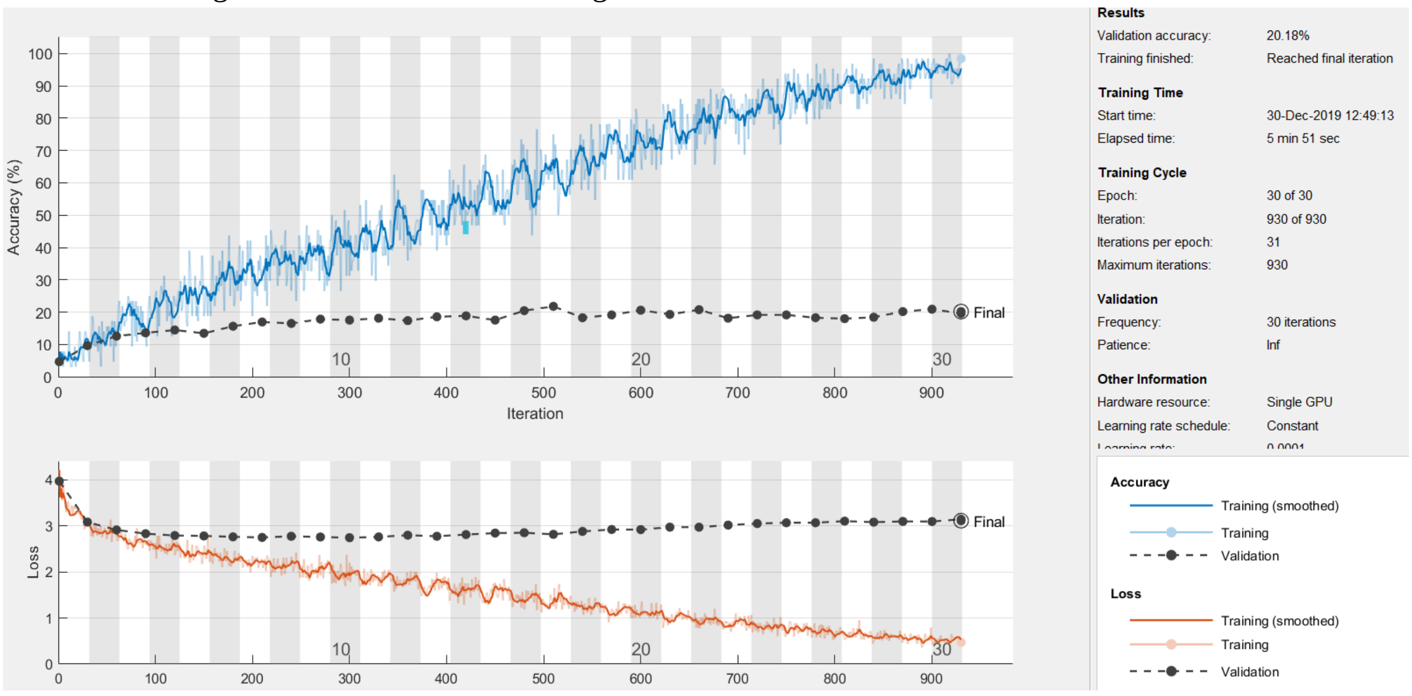
Training Cycle
Epoch: 5 of 5
Iteration: 155 of 155
Iterations per epoch: 31
Maximum iterations: 155

Validation
Frequency: 30 iterations
Patience: Inf

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0001

Figure 1. Validation and loss curves with epoch 5

Secondly, we change `'MaxEpochs'` from 5 to 20. Our network are re-trained. Validation accuracy is increased and it is %17.84. Thirdly, we change `'MaxEpochs'` from 20 to 30. Validation accuracy is increased and it is %20.18. After 200 iterations, loss is increasing. The curves are shown in Figure 2.



Results
Validation accuracy: 20.18%
Training finished: Reached final iteration

Training Time
Start time: 30-Dec-2019 12:49:13
Elapsed time: 5 min 51 sec

Training Cycle
Epoch: 30 of 30
Iteration: 930 of 930
Iterations per epoch: 31
Maximum iterations: 930

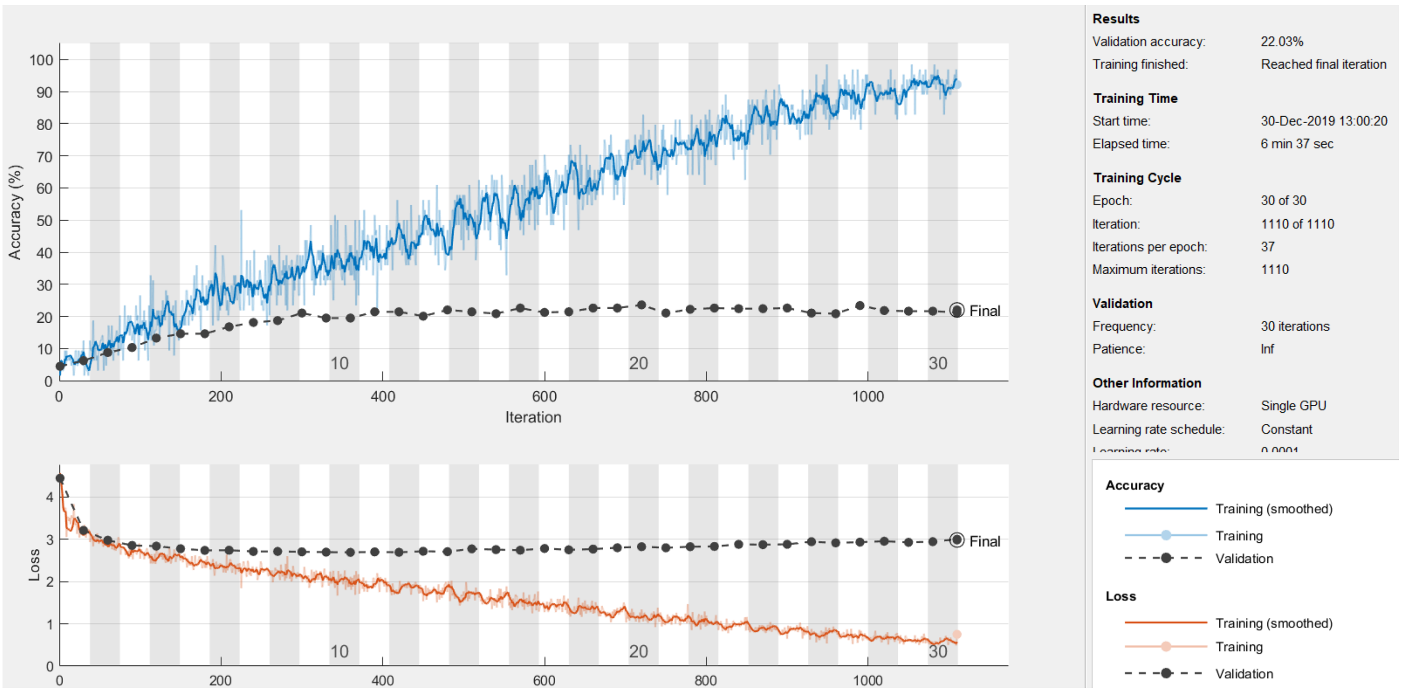
Validation
Frequency: 30 iterations
Patience: Inf

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0001

Figure 2. Validation and loss curves with epoch 30

`'MaxEpochs'` is stabilize at 30. When the learning rate changes to 0.01, our validation accuracy is % 8.04 and loss is dramatically increasing so learning rate is stabilize at 0.0001. When `'MiniBatchSize'` is changed to 128, elapsed time are dramatically increasing. Also when it is changed to 16, elapsed time are dramatically increasing again. Validation accuracy does not change at all. So it is stable at 64. Especially when using GPUs, it is common for power of 2 batch sizes to offer better runtime [2].

Afterwards, we change the ratio of data split. We train the network as %70 training, %15 validation and %15 testing. Validation accuracy is %22.03. Loss curve is increasing after 500 iterations. The curves are shown in Figure 3.



Results

Validation accuracy:	22.03%
Training finished:	Reached final iteration

Training Time

Start time:	30-Dec-2019 13:00:20
Elapsed time:	6 min 37 sec

Training Cycle

Epoch:	30 of 30
Iteration:	1110 of 1110
Iterations per epoch:	37
Maximum iterations:	1110

Validation

Frequency:	30 iterations
Patience:	Inf

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Accuracy
- Training (smoothed)
- Training
- Validation

Loss
- Training (smoothed)
- Training
- Validation

Figure 3. Second data split features

Our next experiment is about data augmentation. Preview the random transformations applied to the first eight images in the image datastore in Figure 4. Figure 5 is shown the last experiment. We use second data split features. Here is the augmented features:

```
    imageAugmenter = imageDataAugmenter( ...
   'RandRotation',[-20,20], ...
   'RandXReflection',true, ...
   'RandXTranslation',[-30 30], ...
   'RandYTranslation',[-30 30], ...
   'RandXScale',[0.9 1.1], ...
   'RandYScale',[0.9 1.1]);
```

And then our validation accuracy is %22.42 and test accuracy is %23.29.

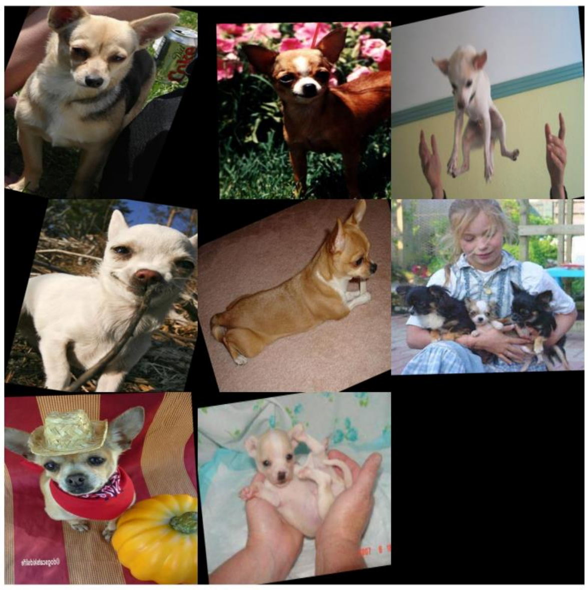
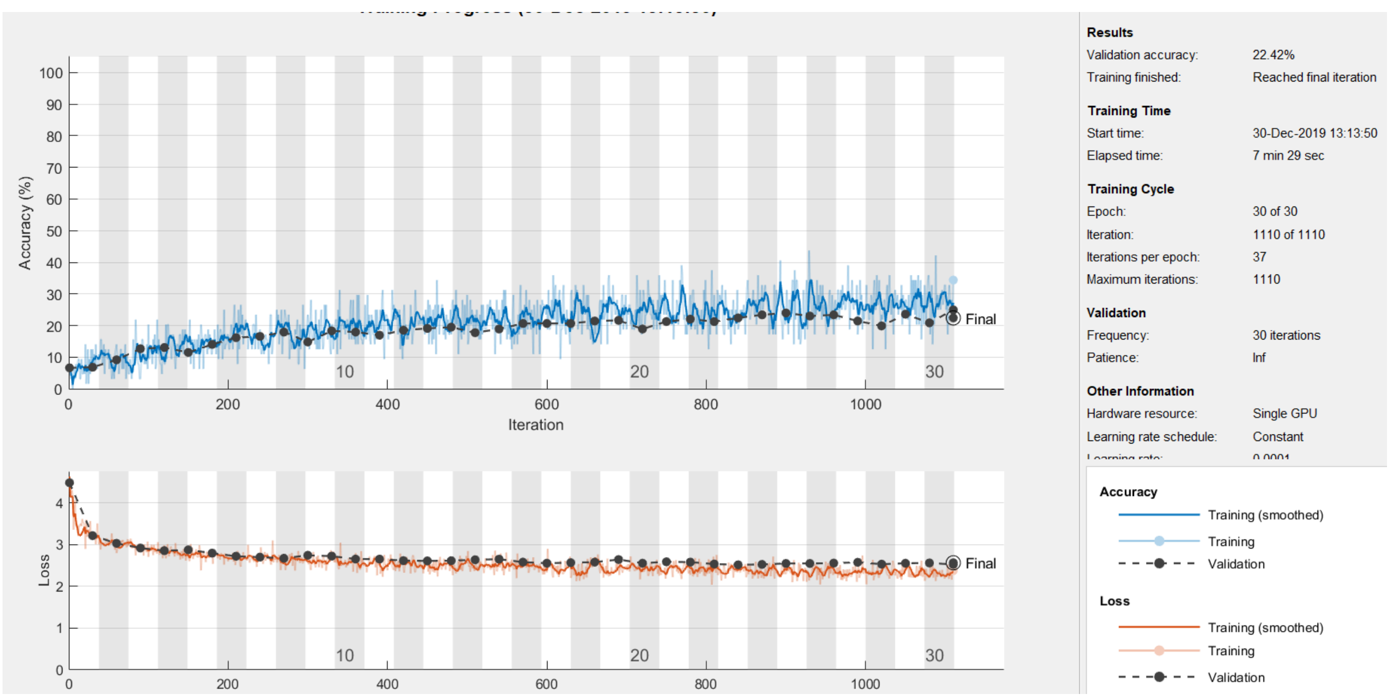


Figure 4. Data Augmentation Examples



Results

Validation accuracy:	22.42%
Training finished:	Reached final iteration

Training Time

Start time:	30-Dec-2019 13:13:50
Elapsed time:	7 min 29 sec

Training Cycle

Epoch:	30 of 30
Iteration:	1110 of 1110
Iterations per epoch:	37
Maximum iterations:	1110

Validation

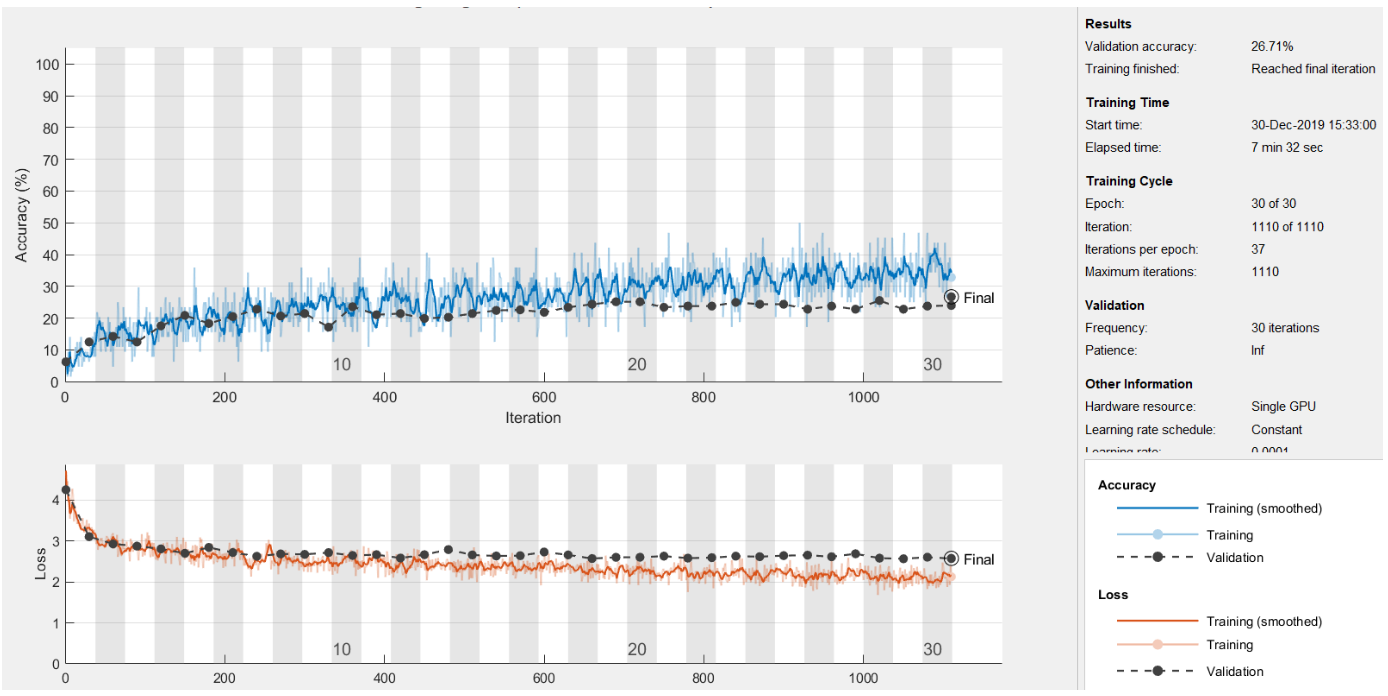
Frequency:	30 iterations
Patience:	Inf

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant

Figure 5. The Experiment with Data Augmentation

Unlike the other loss curves, this experiment loss curve is stabilize. Data augmentation techniques are increasing our validation accuracy. Afterwards we use 'adam' optimizer. Then our validation accuracy is %26.71 and test accuracy is %23.48. Afterwards we use 'rmsprop' optimizer. Then our validation accuracy is %23.2 and test accuracy is %21.72. Figure 6 shows the experiment with 'adam' optimizer and data augmentation.



Results	
Validation accuracy:	26.71%
Training finished:	Reached final iteration
Training Time	
Start time:	30-Dec-2019 15:33:00
Elapsed time:	7 min 32 sec
Training Cycle	
Epoch:	30 of 30
Iteration:	1110 of 1110
Iterations per epoch:	37
Maximum iterations:	1110
Validation	
Frequency:	30 iterations
Patience:	Inf
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Figure 6. The Experiment with `'adam'` Optimizer

We reach the biggest validation ratio in the `'adam'` optimizer. Also changing of data split ratio and data augmentation techniques are increasing our validation accuracy.

To sum up, we have 16 layers. Layers calculates and sets the size of the zero padding so that the layer output has the same size as the input. First convolution layer creates a 2-D convolutional layer with 8 filters of size [3 3]. Second convolution layer creates a 2-D convolutional layer with 16 filters of size [3 3]. And third convolution layer creates a 2-D convolutional layer with 32 filters of size [3 3]. These maxpooling layers create a max pooling layer with pool size [2 2] and stride [2 2]. By doing this, we decrease the size of input image. Also we use three `batchNormalizationLayer` to reduce overfitting and to speed up the network. While we use `dropoutLayer` it is decreasing the validation accuracy because of overfitting. Figure 7 shows the confusion matrix. Here is the network and hyperparameters:

```
layers = [

    imageInputLayer([256 256 3])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
```

```
        reluLayer
        maxPooling2dLayer(2,'Stride',2)

        fullyConnectedLayer(20)
        softmaxLayer
        classificationLayer];

    options = trainingOptions('adam', ...
        'MiniBatchSize',64, ...
        'InitialLearnRate',0.0001, ...
        'MaxEpochs',30, ...
        'Shuffle','every-epoch', ...
        'ValidationData',imageValidation12, ...
        'ValidationFrequency',30, ...
        'Verbose',false, ...
        'Plots','training-progress');
```

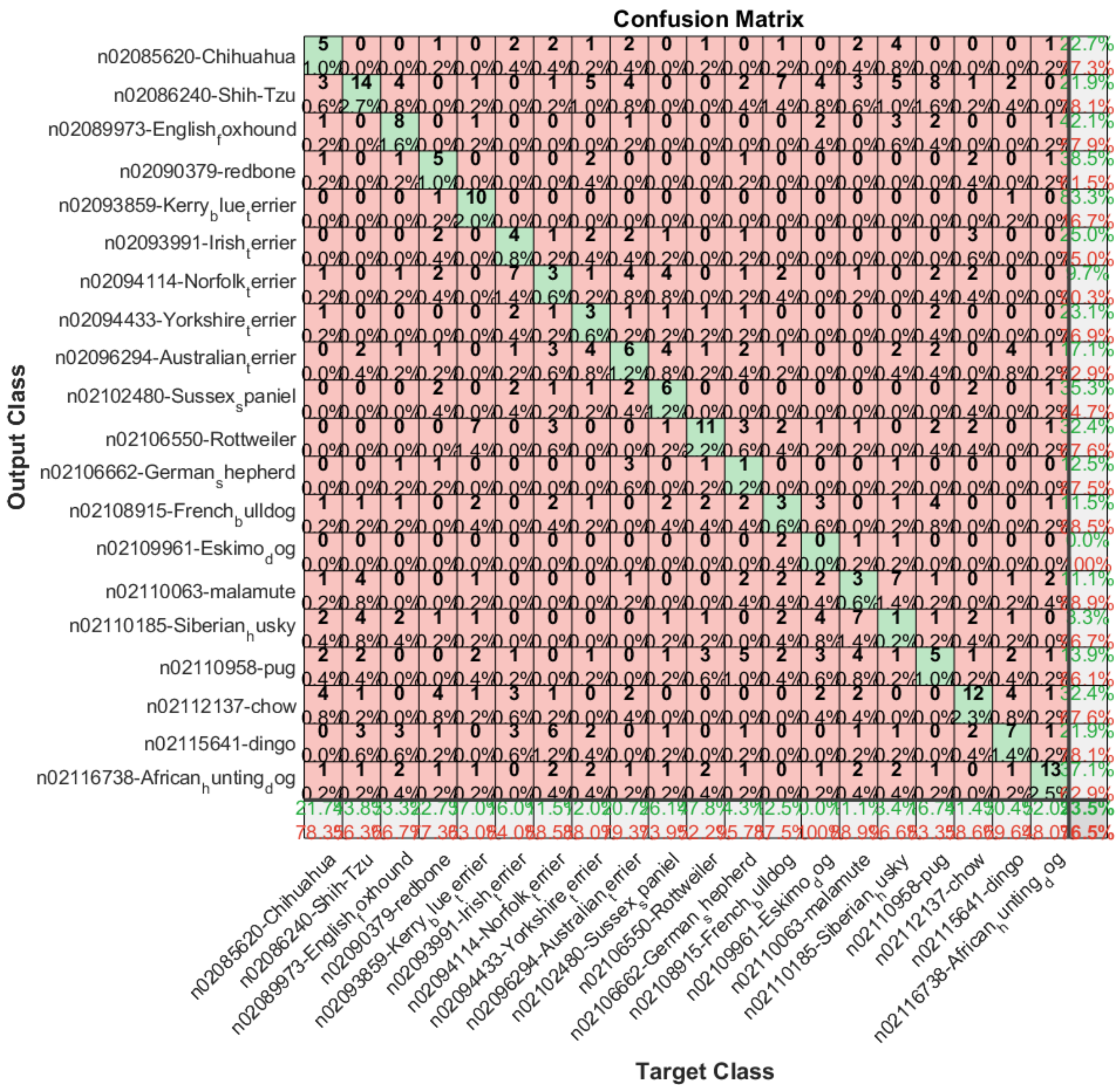


Figure 7. The Confusion Matrix of Our Experiment

PART III.

In the last part, we use pre-trained network ResNet18. When `'MiniBatchSize'` is 128, our device has a warning ‘Out of memory on device’ so we need to lower batch-size. We have to change the value from 128 to 64. Then in Resnet18, we change the last FC layer and Classification Output Layer because we have 20 classes of dogs. We change WeightLearnRateFactor and BiasLearnRateFactor values to 10 for fast learning. Also the input size is 224 x 224 x 3. With data augmentation, our validation accuracy is %80.90 and test accuracy %82.78. The confusion matrix is shown in Figure 8 and validation accuracy and loss curves are shown in Figure 9.

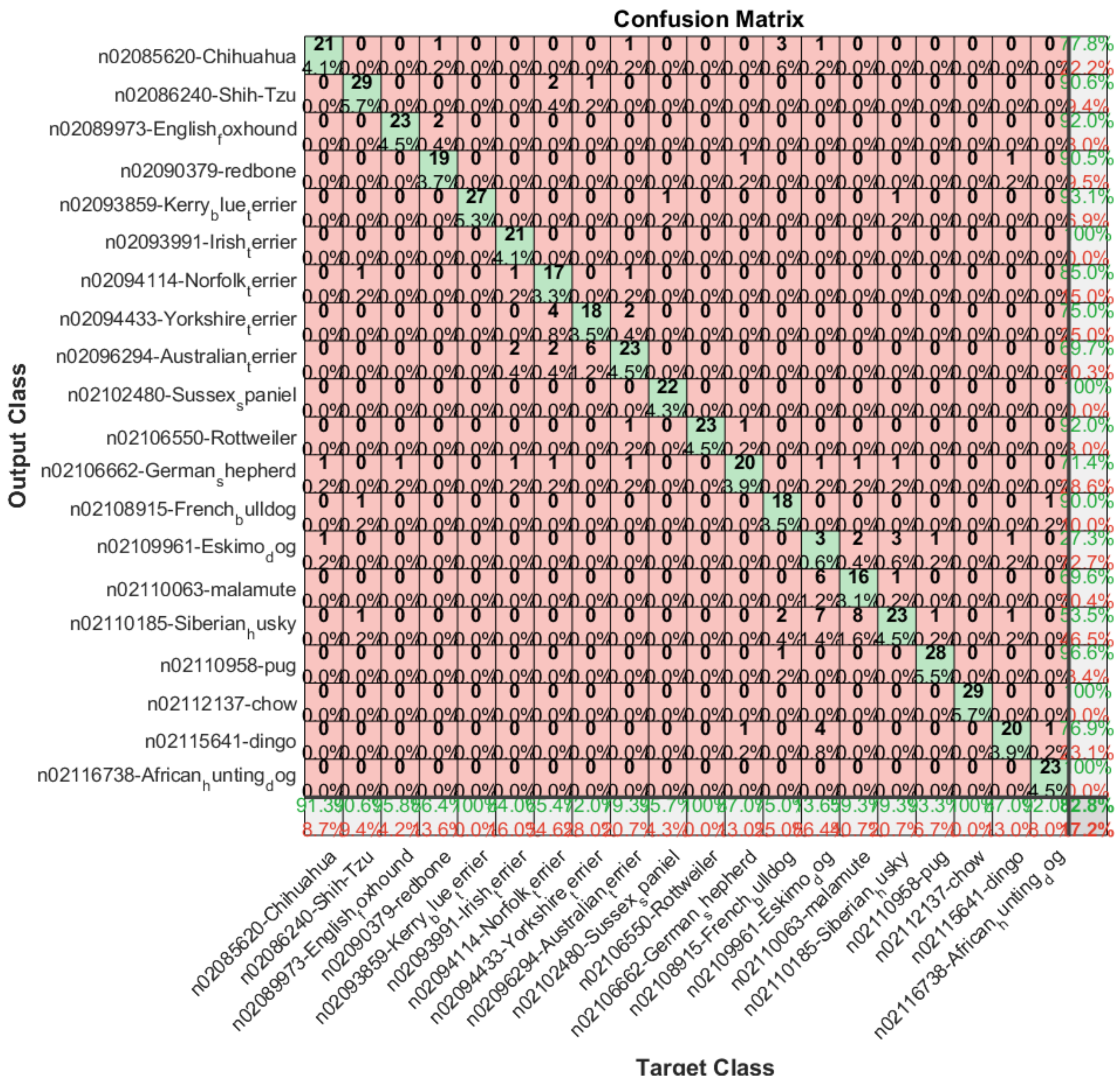
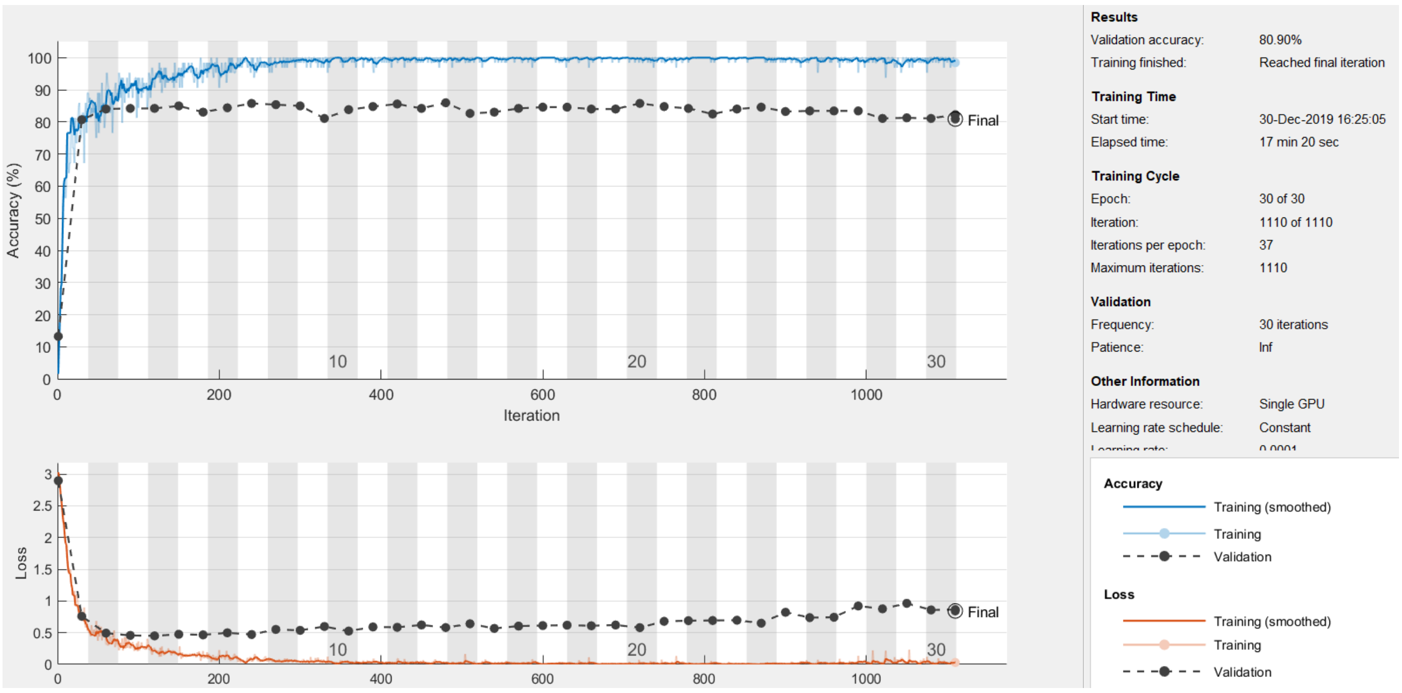


Figure 8. The Confusion Matrix of ResNet18



Results
Validation accuracy: 80.90%
Training finished: Reached final iteration

Training Time
Start time: 30-Dec-2019 16:25:05
Elapsed time: 17 min 20 sec

Training Cycle
Epoch: 30 of 30
Iteration: 1110 of 1110
Iterations per epoch: 37
Maximum iterations: 1110

Validation
Frequency: 30 iterations
Patience: Inf

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0001

Figure 9. Validation and Loss Curves of ResNet18

CONCLUSION

During the experiment, we faced overfitting in Figure 2 and Figure 3 because while training loss is decreasing, validation loss is increasing. We solve this problem by using data augmentation since in Figure 5, there is no overfitting. Also L1 and L2 regularization can be used. The validation accuracy is improved and loss is decreased in Resnet18 markedly because Resnet18 is pre-trained network.

We used cross entropy all of these experiment because it is used to solve multiclass problems.

To improve the performance, we can increase the number of data. If the number of data is increasing, also train, test and validation sets are also increasing. We will have more information about them. Secondly, we can use cross validation method during splitting the data. This method prevents the overfitting and increases the accuracy. Also we can use freeze the some of ResNet18's layers. Freezing the weights of many initial layers can significantly speed up network training. Also we can use early-stopping mechanism in Figure 2 and Figure 3 since after some iterations validation loss is increasing.

Finally, if we compare these two network, In pre-trained network, it has more advantageous than training from scratch because pre-trained network have been trained to work on a lot of different things such as ImageNet, which contains 1.2 million images with 1000 categories. We faced overfitting problem during training from scratch but we don't faced ResNet18. Also we can use another pre-trained networks to compare their results.

REFERENCES

1. http://vision.stanford.edu/aditya86/ImageNetDogs/
2. https://www.deeplearningbook.org/contents/optimization.html