UNIVERSITY OF OULU

521153S –Deep Learning

Final Report

The link of checkpoint file:
https://drive.google.com/open?id=1NwFItevxVtugYNMdXAye
S0uGXoF6A-QE

Group Member
Name: Hao Ban
Student number: 2591928
Name: Yawen Cui
Student number: None

ABBREVIATIONS

std	Standard Deviation
ResNet	Residual Neural Network
FLOPs	Floating Point Operations Per Second
SGD	Stochastic Gradient Descent

TABLE OF CONTENTS

1 INTRODUCTION

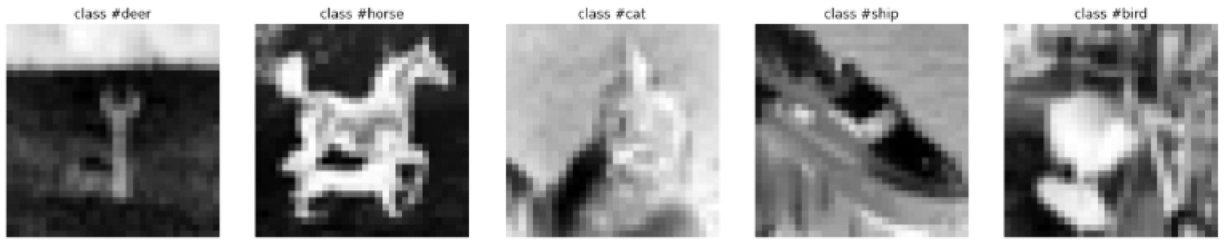
This document is a final report for Deep Learning course. We defined a model and trained based on available dataset which has been divided into training and validation set. In Section 2, the process of data preprocessing is introduced. We defined the network structure and parameters of the model in Section 3 and updated the weight of model based on the training set. Finally, we tested our model on the testing set.

2 DATA PREPROCESSING

In order to obtain a better performance, we made data preprocessing on our dataset, including dataset checking, data standardization, image resizing, data augmentation and etc.

2.1 Creating Dataset and Checking

The dataset includes 45,000 images grayscale 32x32 and corresponding labels. We define our training data with 36,000 images and validation data with 9000 images. In order to use our data set, we define a class named *MiniImageNetDataset*, which is similar as class in assignment 4. In order to check the dataset, we also print random images in five different classes. The result from training set is shown in Figure 1.



```
total number of training set: 36000
numer of images for class airplane: 4000
numer of images for class automobile: 4000
numer of images for class bird: 4000
numer of images for class cat: 4000
numer of images for class deer: 4000
numer of images for class dog: 4000
numer of images for class frog: 4000
numer of images for class horse: 4000
numer of images for class ship: 4000
```

Figure 1. Data Example in Training Dataset

2.2 Data Augmentation

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks [1].

In our project, we use the same method as assignment 4, and build our custom data augmentation. We can define a custom data transformation for data augmentation namely *RandomWindowDrop*. There are 50% and 50% chance that it will randomly cut a square window inside our image or horizontally flip the image, which has been shown in Figure 2.

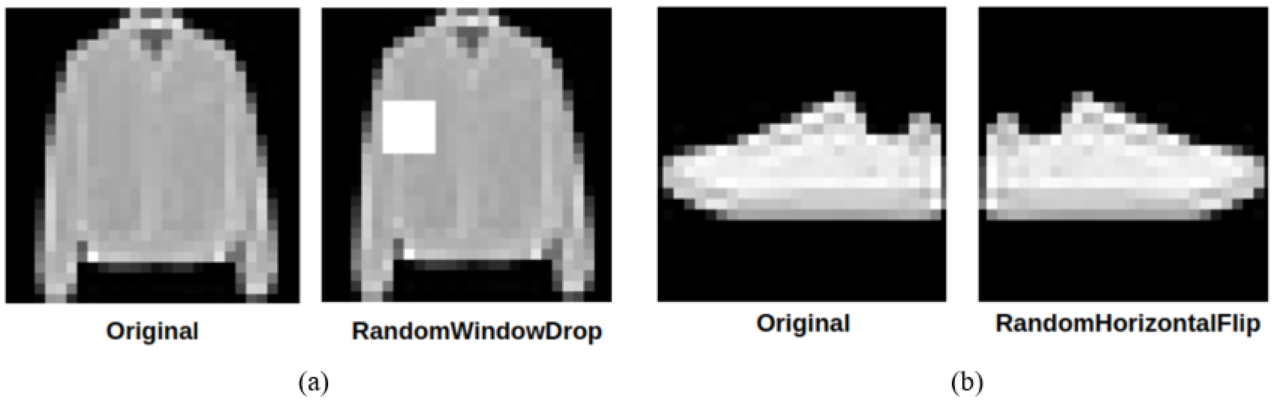


Figure 2. Data Augmentation by Random Window Drop

2.3 Loading Data

When loading training data and validation data, dataset need to be normalized and transformed in order to obtain a better performance.

At first, we use function transforms.ToTensor() to convert image into torch.FloadTensor and normalize the value between [0.0, 1.0]. And then, we define *mean* and *std* value as 0.5 to normalize the data set in order to make the value between [-1.0, 1.0]. Except this, the size of images has been resized as 224x224 in order to fit our model. For the process of resizing, we define default bilinear interpolation to implement it.

3 DEFINING MODEL

According to the result from ImageNet competition, we prefer to use ResNet [2] proposed by Kaiming He to build our model. ResNet solves the problem of vanishing gradients by skipping over layers.

3.1 Network Architecture

We select ResNet 18 as basic model with BasicBlock as [2, 2, 2, 2]. Each block has two convolutional layers, batch normalization layers and one Relu layer. The model network architecture has been shown in Table 1. In order to fit our training data, the number of labels in fully connected layer has been changed into 9. We also take Figure 3 as an instance to illustrate how skipping over layers work in our model.

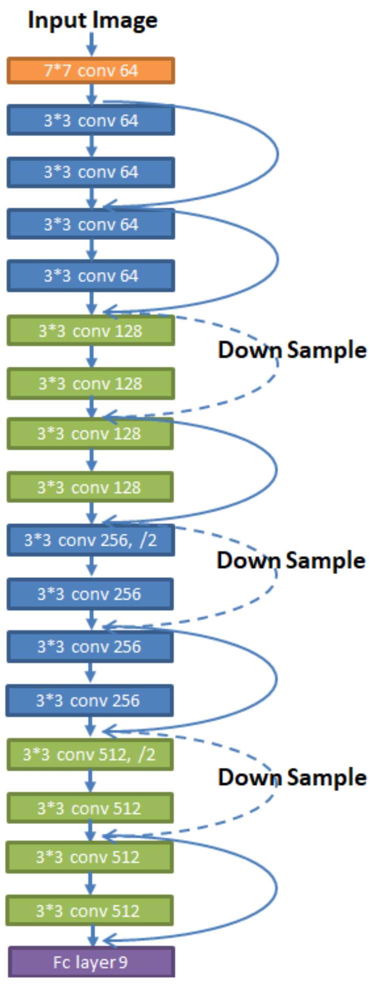


Figure 3. Residual Network 18 Layers Architecture

Table 1. Model Network Architecture

Layer name	*Output size*	*Model ResNet18*
Input layer	224*224	
Conv1	112*112	7*7, 64, stride 2
Conv2_x	56*56	3*3, 64, stride 1
Conv3_x	28*28	3*3, 128, stride 1
Conv4_x	14*14	3*3, 256, stride 1
Conv5_x	7*7	3*3, 512, stride 1
Output Size	1*1	
FLOPs		$1.8*10^9$

3.2 Model Parameters

For the hyperparameters in our model, we build *SGD* optimizer with learning rate=0.01, momentum=0.9, weight decay = 1e-10 as the original value and set time decay of learning rate. After 30 epochs, we set learning rate=0.005. Batch size is limited by the size of GPU, and it also will modify along with different models. In our model, we set batch size=64 as default value.

3.3 Training and Validation

Once hyperparameters were set, we trained this model for 50 epochs. And after 30 epochs, we modified the optimizer and changed learning rate to 0.005. Tensorboard has been used in this part to visualize the process of training.

After 50 epochs finished, the curve of accuracy for training data and validation data is shown in Figure 4. And the curve of loss for training data and validation data is shown in Figure 5.

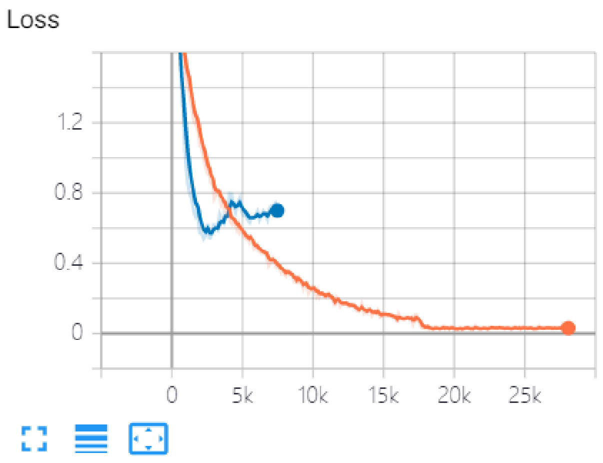


Figure 4. The Curve of Loss for Training 50 Epochs

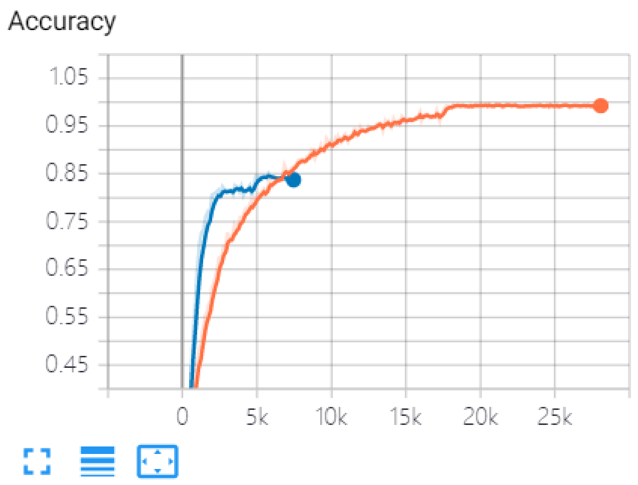


Figure 5. The Curve of Accuracy for Training 50 Epochs

As you can see, the accuracy of the training set after training for multiple epochs is close to 100%, and the accuracy of the validation set reaches 85%. In addition, at the discontinuous points of the curve, the accuracy of the curve is obvious improved by adjusting the learning rate.

4 TESTING MODEL AND DISCUSSION

It's obvious to see that the model has an overfitting problem. In order to solve this problem, we also tried other models, such as *ResNet* 50. And we also tried to adjust the hyper parameters such as learning rate, epochs, weight decay and batch size. They did not achieve good results and the training process took a long time.

Therefore, we processed the test set with the existing model and obtained an accuracy of 84.06%, which has met the basic requirements of 82.5%. In the future, I hope to use the learned methods and more experience to optimize the model and reduce overfitting.

5 REFERENCES

[1] 1000x Faster Data Augmentation by Berkeley Artificial Intelligence Research, https://bair.berkeley.edu/blog/2019/06/07/data_aug/

[2] He, Kaiming , et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) IEEE Computer Society, 2016.