Mini Assignment: Implementation of a Small VGG Network using Pytorch

A ConvNet was implemented in the file *convnet_pytorch.py*. Training and testing procedures are implemented in the file *train_convnet_pytorch.py*. The ConvNet architecture is a small version of the popular VGG network, and is depicted in Table 1.

Name	Kernel	Stride	Padding	Channels In/Out
conv1	3×3	1	1	3/64
maxpool1	3×3	2	1	64/64
conv2	3×3	1	1	64/128
maxpool2	3×3	2	1	128/128
conv3_a	3×3	1	1	128/256
conv3_b	3×3	1	1	256/256
maxpool3	3×3	2	1	256/256
conv4_a	3×3	1	1	256/512
conv4_b	3×3	1	1	512/512
maxpool4	3×3	2	1	512/512
conv5_a	3×3	1	1	512/512
conv5_b	3×3	1	1	512/512
maxpool5	3×3	2	1	512/512
linear	–	-	-	512/10

Table 1. Specification of ConvNet architecture. All *conv* blocks consist of 2D-convolutional layer, followed by Batch Normalization layer and ReLU layer.

The ConvNet model was trained and tested on images from the cifar-10 dataset (http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz). This dataset contains images of 10 categories. Adam optimizer with default learning rate and PyTorch default initialization parameters were used. The model created with pytorch version 1.10.0 and trained in Google Colab.

After every 500 steps of training, the predictive ability of the model (accuracy = predicted image category / true image category) was tested. The results of this are depicted in Figure 1. Furthermore, the error loss is depicted in Figure 2.

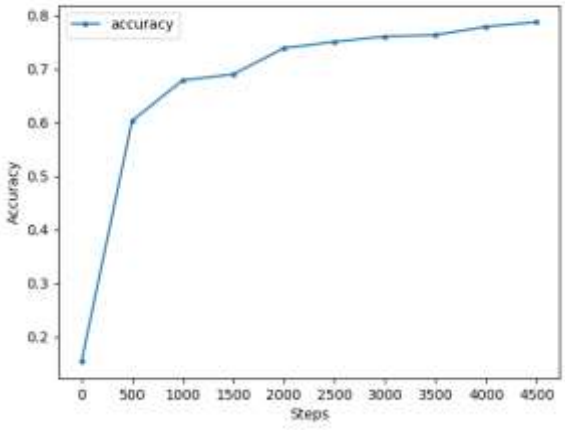

Figure 1: Accuracy of the ConvNet model.

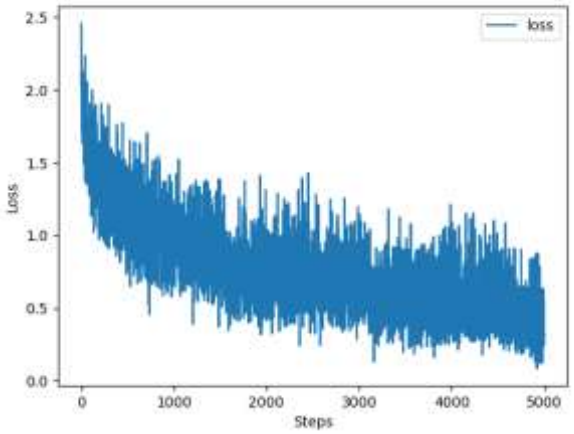

Figure 2: Loss curve

During 5000 steps, the accuracy of the trained model has increased to 0.7875 on the test set.