**CSE-6363 Machine Learning, Spring 2022**

**Programming Assignment 2**

For this programming assignment, I have implemented Lenet 5 CNN using PyTorch.

**Brief Report of the process and results:**

1. We first import all the necessary libraries.
2. We'll create a variable device that will store either 'cpu' or 'gpu' depending on the type of training we're doing.
3. We will download the necessary dataset for training and validation in this phase.

After that, we'll have the MNIST dataset. We resize our images to (32x32) and then convert them to tensors, which will scale them to the [0, 1] range automatically. In addition, we set the batch size to 64 and shuffle our data.

1. Extract one batch from the training set, and we can see that we have 64 images of shape (1x32x32) and 64 labels if we look at the shape.
2. Now comes the fun part. The LeNet5 model is being built. We begin by creating a LeNet5 class and a \_\_init\_\_ function.

Splitting the convolutional and linear layers is the simplest and most elegant technique to design this model.

To do so, we'll make two sequential models, one called self.convolutional layer and the other self.linear layer. With three activation functions, we'll generate three convolutional layers and two average pooling layers for the convolutional layer.

We'll define two linear layers for the linear layer, with a tanh function as the activation function after the first. We'll use the softmax function in the forward function directly.

We first use a self.convolutional\_layer on the image in the forward function, then flatten the result and feed it into the self.

Linear layer, and the softmax is applied to the result of the linear layer.

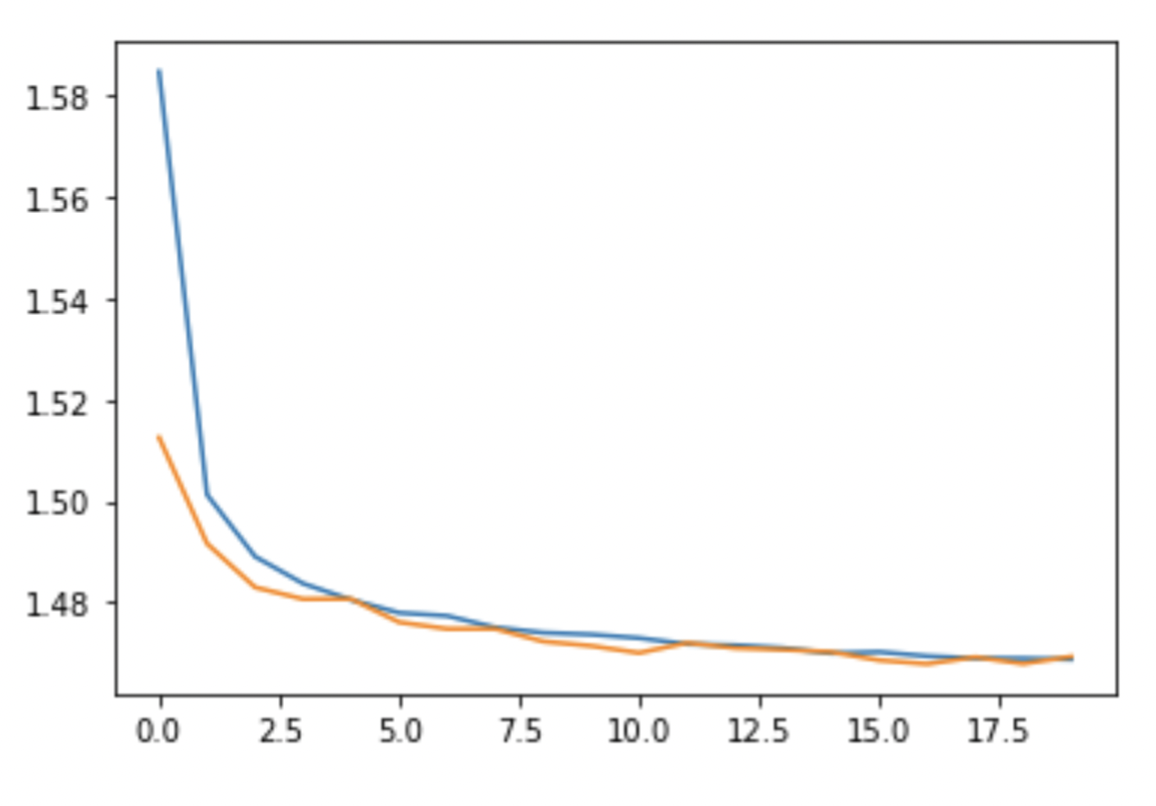
1. Call the model we just made and tell it to work with the device we specified at the start.
2. We use Adam for the optimizer and CrossEntropyLoss for the loss.
3. We'll put it through 20 iterations of training. We go over all of the training photos one by one, making sure they work with the device we specified at the start.

Conduct the forward propagation step, calculate the loss, and then do the backpropagation step. Following the training phase, we shall proceed to the evaluation phase as usual.

We go through all of the testing photos and make predictions using our model. Every time our model predicts properly, we add 1 to the total and use this to calculate the accuracy. Every 5th epoch, the training loss, validation loss, and validation accuracy will be printed.

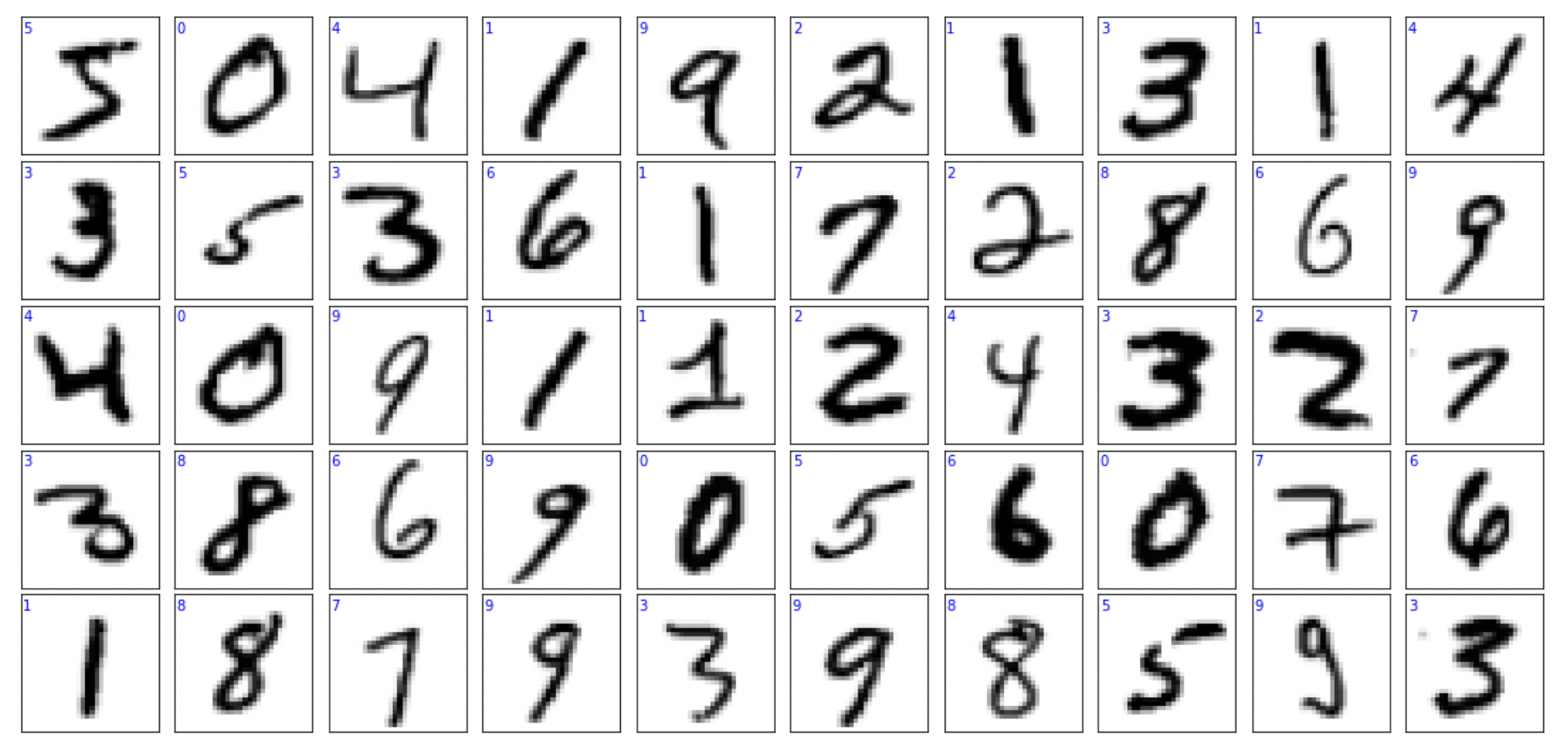
1. We'll plot the training and validation losses to have a better idea of how our model performed.
2. We'll also take one batch from the testing set and extract the photos and labels from it.
3. We next apply our model to that batch, or those 64 images, to make predictions.
4. Using this single line for loop, we extract all of the photos into a list. The data is converted to numpy. Make a list of classes that you can find on the web.
5. We plot 50 photos with their matching labels after iterating over them 50 times. We'll color the label blue if our model correctly predicted the class, and red if it didn't.

**An analysis of results:**

****

The training loss plateaus(blue line), while the validation loss (orange line) has little bumps from time to time (increased values). Overall, I thought the performance was satisfactory. The 16th epoch produced the best results (on the validation set).

We exhibited a set of values from the validation set, together with the predicted label and the probability that the network gives to that label, to evaluate our model's predictions (in other words, how confident the network is in the prediction).



As we can see in the output that all the labels (on the top left) are blue colored and none is red, it means that our model correctly predicted all the 50 classes here, and none was predicted incorrectly.

**References:**

* <https://www.kaggle.com/competitions/digit-recognizer/code>
* <https://towardsdatascience.com/implementing-yann-lecuns-lenet-5-in-pytorch-5e05a0911320>
* <https://www.youtube.com/watch?v=HPXnJmO8yQ4&t=569s>
* <https://www.youtube.com/watch?v=ijaT8HuCtIY>
* <https://github.com/krishna-tx/mnist-pytorch/blob/master/mnist_cnn.ipynb>
* <https://github.com/maticvl/dataHacker/blob/master/pyTorch/010_leNet.ipynb>