Sami Emre Erdogan
Student ID: 2019280513

Assignment 4 Pytorch

MLP

1. **With and without dropout layer**

In my assignment I have modelled my network with and without dropout layer in MLP. It has 2 hidden layers My Network and model are shown below.

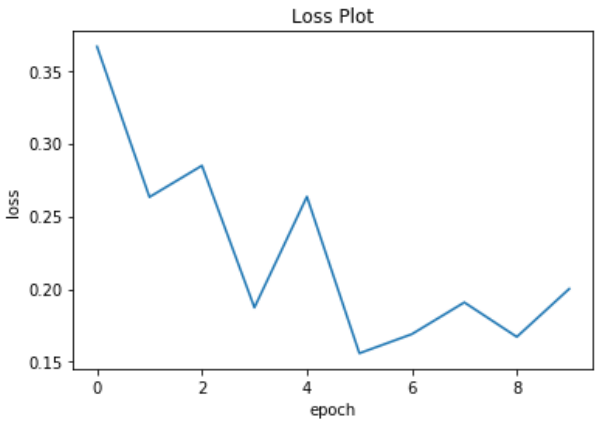
```
class MLP(nn.Module):
    def __init__(self, n_features, n_classes):
        super(MLP, self).__init__()
        self.layer1 = nn.Linear(n_features, 128)
        self.layer2 = nn.Linear(128, 128)
        self.layer3 = nn.Linear(128, n_classes)
```

With Dropout MLP

```
def forward(self, x, training=True):
    # a neural network with 2 hidden layers
    # x -> FC -> relu -> dropout -> FC -> relu -> dropout -> FC -> output
    x = F.relu(self.layer1(x))
    x = F.dropout(x, 0.5, training=training)
    x = F.relu(self.layer2(x))
    x = F.dropout(x, 0.5, training=training)
    x = self.layer3(x)
    return x
```

When I ran this code, the results are obtained as shown below:
batch_size = 250
epoch_num = 10
lr = 0.001
disp_freq = 20

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(0.2002)

Result
Test accuracy:  tensor(97.2000)
loss:  tensor(0.2002)
```



It performed significantly well with an accuracy rate of 97.2000 and loss of 0.2002.
Now let's check without the dropout layers and compare.

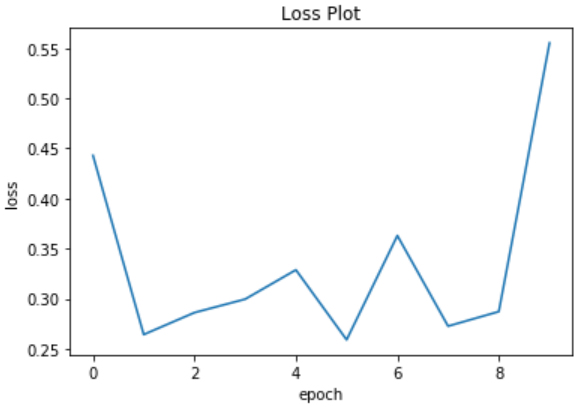
Sami Emre Erdogan
Student ID: 2019280513

Without Dropout Layer MLP

batch_size = 250
epoch_num = 10
lr = 0.001
disp_freq = 20

```
def forward(self, x, training=True):
    # a neural network with 2 hidden layers
    # x -> FC -> relu -> FC -> relu -> FC -> output
    x = F.relu(self.layer1(x))
    x = F.relu(self.layer2(x))
    x = self.layer3(x)
    return x
```

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(0.5551)

_________________________________________________________________
Result
Test accuracy:  tensor(96.4000)
loss:  tensor(0.5551)
```



Through my observations when I used dropout layers for MLP it gave slightly better result and less loss spiking between epochs.
Test Accuracy rate: 96.4000
Loss: 0.5551

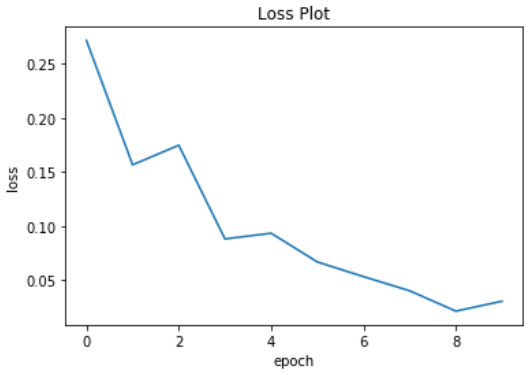
Sami Emre Erdogan
Student ID: 2019280513

2. Learning Rate Comparison MLP

Next, I have compared different learning rates by keeping the rest the same so we can make better comparison. (Dropout layers used)

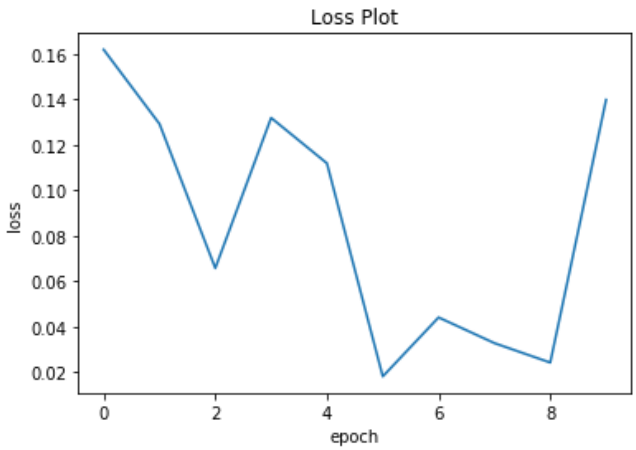
lr = 0.001

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(0.0305)

Result
Test accuracy:  tensor(97.6000)
loss:  tensor(0.0305)
```



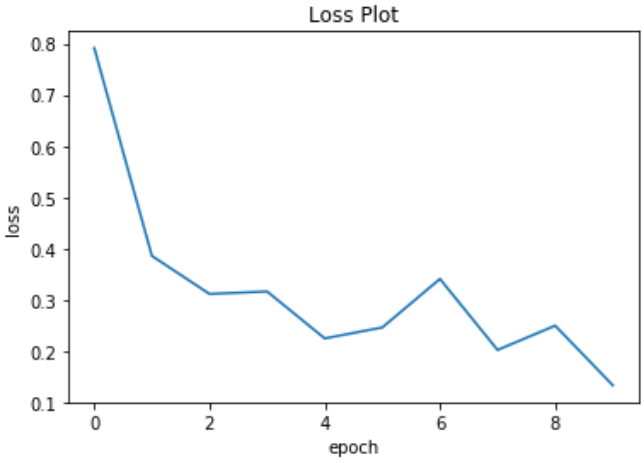
lr = 0.01

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(0.1399)

Result
Test accuracy:  tensor(96.)
loss:  tensor(0.1399)
```



lr = 0.0001

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(0.1325)

Result
Test accuracy:  tensor(95.2000)
loss:  tensor(0.1325)
```



When look at the results we can see that learning rate of 0.001 gave us the most accurate results among these 3.

Sami Emre Erdogan
Student ID: 2019280513

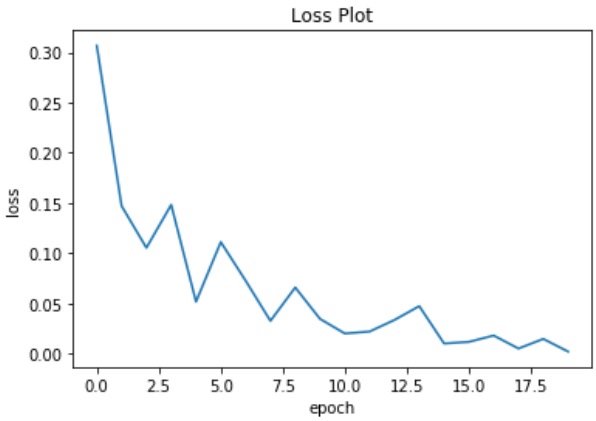
3. Epoch size 10 vs 20 MLP

Since I have done with the results with epoch size 10 I will just show he results with epoch 20. Also, I will be using the same layers I gave which contained the dropout layers so we can compare the two.

Epoch 10
Test Accuracy: 97.2000
loss of 0.2002.

Epoch 20

```
epoch: 19 - batch: 0/220
epoch: 19 - batch: 20/220
epoch: 19 - batch: 40/220
epoch: 19 - batch: 60/220
epoch: 19 - batch: 80/220
epoch: 19 - batch: 100/220
epoch: 19 - batch: 120/220
epoch: 19 - batch: 140/220
epoch: 19 - batch: 160/220
epoch: 19 - batch: 180/220
epoch: 19 - batch: 200/220
loss:  tensor(0.0023)

Result
Test accuracy:  tensor(98.)
loss:  tensor(0.0023)
```



Increasing the epoch has also increased our test accuracy and significantly lowered our loss.

Sami Emre Erdogan
Student ID: 2019280513

CNN

1. **With and without dropout layer CNN**

In my assignment I have modelled my network with and without dropout layer in CNN. The autoencoder has 3 con layers and 3 deconv layers (transposed conv). All layers but the last have ReLu

```
class CNN(nn.Module):
    def __init__(self, n_classes):
        super(CNN, self).__init__()
        # conv layers: (in_channel size, out_channels size, kernel_size, stride, padding)
        self.conv1 = nn.Conv2d(1, 32, 5, stride=1, padding=2)
        self.conv2 = nn.Conv2d(32, 16, 5, stride=1, padding=2)
        self.conv3 = nn.Conv2d(16, 8, 5, stride=1, padding=2)

        # max pooling (kernel_size, stride)
        self.pool = nn.MaxPool2d(2, 2)

        # fully conected layers:
        self.layer1 = nn.Linear(4*4*8, 64)
        self.layer2 = nn.Linear(64, 64)
        self.layer3 = nn.Linear(64, n_classes)
```

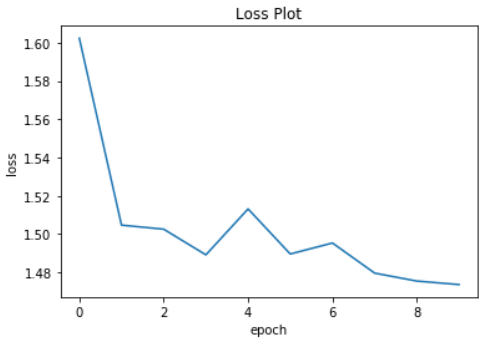
With Dropout Layer CNN

```
def forward(self, x, training=True):
    # the autoencoder has 3 con layers and 3 deconv layers (transposed conv). All layers but the last have ReLu
    # activation function
    x = F.relu(self.conv1(x))
    x = self.pool(x)
    x = F.relu(self.conv2(x))
    x = self.pool(x)
    x = F.relu(self.conv3(x))
    x = self.pool(x)
    x = x.view(-1, 4 * 4 * 8)
    x = F.relu(self.layer1(x))
    x = F.dropout(x, 0.5, training=training)
    x = F.relu(self.layer2(x))
    x = F.dropout(x, 0.5, training=training)
    x = self.layer3(x)
    return x
```

When I ran this code, the results are obtained as shown below:
batch_size = 250
epoch_num = 10
lr = 0.001
disp_freq = 20

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(1.4736)
```

```
Result
Test accuracy:  tensor(99.2000)
loss:  tensor(1.4736)
```



T he test accuracy that I obtained was the best result in my report with an impressive 99.2000 accuracy. The validation loss was also really good.

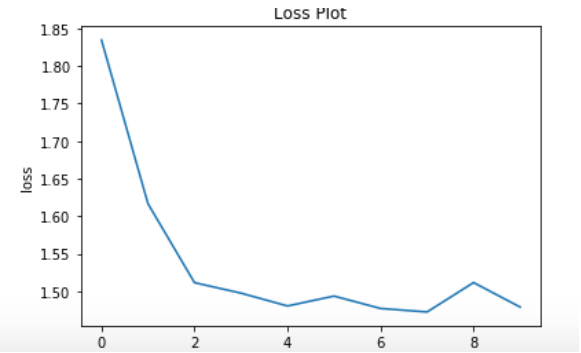
Sami Emre Erdogan
Student ID: 2019280513

Without Dropout Layer CNN

batch_size = 250
epoch_num = 10
lr = 0.001
disp_freq = 20

```
def forward(self, x, training=True):
    # the autoencoder has 3 con layers
    # activation function
    x = F.relu(self.conv1(x))
    x = self.pool(x)
    x = F.relu(self.conv2(x))
    x = self.pool(x)
    x = F.relu(self.conv3(x))
    x = self.pool(x)
    x = x.view(-1, 4 * 4 * 8)
    x = F.relu(self.layer1(x))
    x = F.relu(self.layer2(x))
    x = self.layer3(x)
    return x
```

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(1.4790)

Result
Test accuracy:  tensor(97.6000)
loss:  tensor(1.4790)
```



The testing accuracy and validation loss performed really well when the dropout layer was used but not as good as when dropout layer was used.

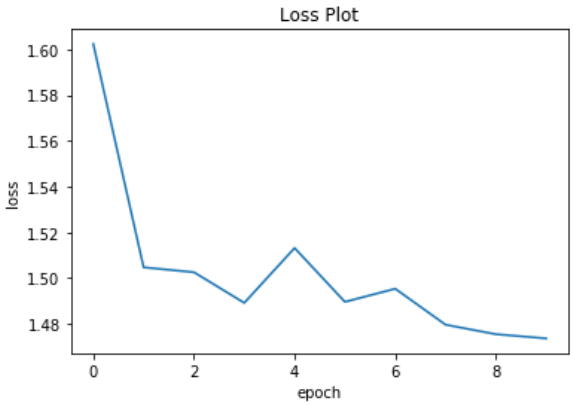
Sami Emre Erdogan
Student ID: 2019280513

2. Learning Rate Comparison CNN

Next, I have compared different learning rates by keeping the rest the same so we can make better comparison. (Dropout layers used)

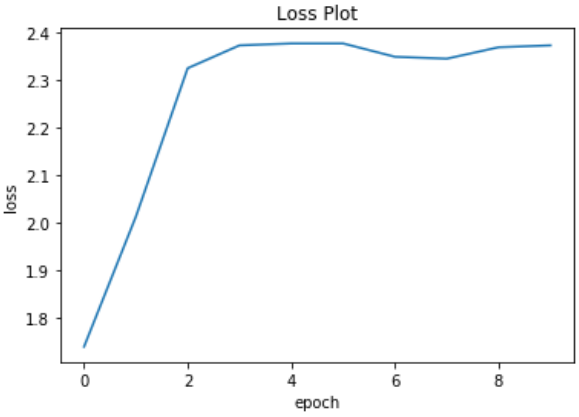
lr = 0.001

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(1.4736)

Result
Test accuracy:  tensor(99.2000)
loss:  tensor(1.4736)
```



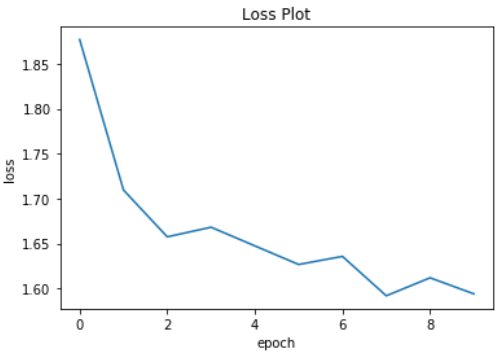
lr = 0.01

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(2.3731)

Result
Test accuracy:  tensor(8.)
loss:  tensor(2.3731)
```



lr = 0.0001

```
epoch: 9 - batch: 0/220
epoch: 9 - batch: 20/220
epoch: 9 - batch: 40/220
epoch: 9 - batch: 60/220
epoch: 9 - batch: 80/220
epoch: 9 - batch: 100/220
epoch: 9 - batch: 120/220
epoch: 9 - batch: 140/220
epoch: 9 - batch: 160/220
epoch: 9 - batch: 180/220
epoch: 9 - batch: 200/220
loss:  tensor(1.5940)

Result
Test accuracy:  tensor(87.6000)
loss:  tensor(1.5940)
```



When look at the results we can see that learning rate of 0.001 gave us the most accurate and the best result. However lowering the lr to 0.0001 dropped the testing accuracy rate significantly to 87.6000

Sami Emre Erdogan
Student ID: 2019280513

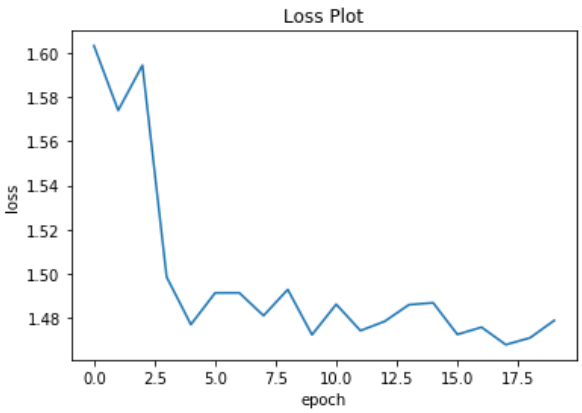
3. Epoch size 10 vs 20 CNN

Since I have done with the results with epoch size 10 I will just show he results with epoch 20. Also, I will be using the same layers I gave which contained the dropout layers so we can compare the two.

Epoch 10
Test Accuracy: 99.2000
loss of 1.4736.

Epoch 20

```
epoch: 19 - batch: 40/220
epoch: 19 - batch: 60/220
epoch: 19 - batch: 80/220
epoch: 19 - batch: 100/220
epoch: 19 - batch: 120/220
epoch: 19 - batch: 140/220
epoch: 19 - batch: 160/220
epoch: 19 - batch: 180/220
epoch: 19 - batch: 200/220
loss:  tensor(1.4790)

_____________________________________
Result
Test accuracy:  tensor(98.8000)
loss:  tensor(1.4790)
```



In this case increasing the epoch to 20 hasn't increased the accuracy rate but decreased it slightly compared to epoch 10.

4. Final Thoughts of CNN and MLP with TensorFlow Compared to previous Homework

Firstly when I was training the data I have seen a significant improvement when it came to with pytorch and just numpy implementation from previous homework. This Homework gave me more accurate results while performing much faster.

Secondly, When I compare the MLP and CNN implementation with pytorch. The results obtained was significantly similar but the speed of MLP while doing training was really faster than CNN when it came to training. If you have a poor CPU or GPU I would suggest anyone to use while if you have a powerful CPU or GPU CNN task shouldn't be a problem because I used my Macbook Air to compute the tasks and I had to wait a long time for CNN.