https://github.com/Christian-Martens-UNCC/ECGR-4105/tree/main/Homework_6-ANN_%26_CNN

Problem 1:

A) 50% accuracy after 300 epochs and an hour and 25 minute of training is not terrible considering the data that the model was trained on is very pixelated/compressed. I tried a separate training session with a substantially higher batch size and learning rate, but the loss of the function was very high since the network couldn't properly derive any meaning from the very random batches. I also tried a training session with a low batch size and higher learning rate in which the training loss reduced down to nearly 0, but the evaluation accuracy was overall lower than the shown accuracy by about 0.10 which clearly shows that overfitting occurred during the training session.

```
model = nn.Sequential(nn.Linear(3072, 512),
                      nn.Tanh(),
                      nn.Linear(512, 10),
                      nn.LogSoftmax(dim=1))
```

```
Epoch 300, Duration = 18.015 seconds, Loss: 0.84595

Total Training Time = 5214.901 seconds
Average Training Time per Epoch = 17.383 seconds

Accuracy: 0.495, Duration = 3.295 seconds
```

B) The size of the model is larger, though not by much, compared to 1.a. Additionally, this model was less accurate than the previous model despite having a reduced loss which means that this model was overfitted.

```
model_2 = nn.Sequential(nn.Linear(3072, 512),
                        nn.Tanh(),
                        nn.Linear(512, 128),
                        nn.Tanh(),
                        nn.Linear(128, 32),
                        nn.Tanh(),
                        nn.Linear(32, 10),
                        nn.LogSoftmax(dim=1))
```

```
Epoch 300, Duration = 18.036 seconds, Loss: 0.37906

Total Training Time = 5434.326 seconds
Average Training Time per Epoch = ~18.114 seconds

Accuracy: 0.458, Duration = 2.648 seconds
```

Problem 2:

A) The CNN performed better than the model in 1.a in nearly every way. The training was faster by about 20% and the resulting validation accuracy was improved by 15%.

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size = 3, padding = 1)
        self.act1 = nn.Tanh()
        self.pool1 = nn.MaxPool2d(2)
        self.conv2 = nn.Conv2d(16, 8, kernel_size = 3, padding = 1)
        self.act2 = nn.Tanh()
        self.pool2 = nn.MaxPool2d(2)
        self.fc1 = nn.Linear(8 * 8 * 8, 32)
        self.act3 = nn.Tanh()
        self.fc2 = nn.Linear(32, 10)

    def forward(self, x):
        out = self.pool1(self.act1(self.conv1(x)))
        out = self.pool2(self.act2(self.conv2(out)))
        out = out.view(-1, 8 * 8 * 8)
        out = self.act3(self.fc1(out))
        out = self.fc2(out)
        return out
```

```
Epoch 300, Duration = 14.248 seconds, Loss: 0.94016

Total Training Time = 4040.299 seconds
Average Training Time per Epoch = 13.468 seconds

Loader: train, Accuracy: 0.674, Duration = 11.782 seconds
Loader: val, Accuracy: 0.655, Duration = 2.308 seconds
```

B) It does not appear that the reduction in accuracy was a result of overfitting but rather a result of a small learning rate (all learning rates and batch sizes were kept standard across all tests for comparison purposes). With a larger number of epochs or a higher learning rate, it is possible that overfitting would've occurred with this model. That being said, this model did not perform as well as the CNN in 2.a as its accuracy was not as high given the same epoch and batch sizes.

```
class Net_2(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size = 3, padding = 1)
        self.act1 = nn.Tanh()
        self.pool1 = nn.MaxPool2d(2)
        self.conv2 = nn.Conv2d(16, 8, kernel_size = 3, padding = 1)
        self.act2 = nn.Tanh()
        self.pool2 = nn.MaxPool2d(2)
        self.conv3 = nn.Conv2d(8, 4, kernel_size = 3, padding = 1)
        self.act3 = nn.Tanh()
        self.pool3 = nn.MaxPool2d(2)
        self.fc1 = nn.Linear(4 * 4 * 4, 32)
        self.act4 = nn.Tanh()
        self.fc2 = nn.Linear(32, 10)

    def forward(self, x):
        out = self.pool1(self.act1(self.conv1(x)))
        out = self.pool2(self.act2(self.conv2(out)))
        out = self.pool3(self.act3(self.conv3(out)))
        out = out.view(-1, 4 * 4 * 4)
        out = self.act4(self.fc1(out))
        out = self.fc2(out)
        return out
```

```
Epoch 300, Duration = 13.438 seconds, Loss: 1.17562

Total Training Time = 4124.658 seconds
Average Training Time per Epoch = 13.749 seconds

Loader: train, Accuracy: 0.583, Duration = 10.645 seconds
Loader: val, Accuracy: 0.573, Duration = 2.113 seconds
```