Stanly Gomes
Student ID: 801118166

Homework 6

GitHub Repository Link: https://github.com/NaraPvP/IntroToML

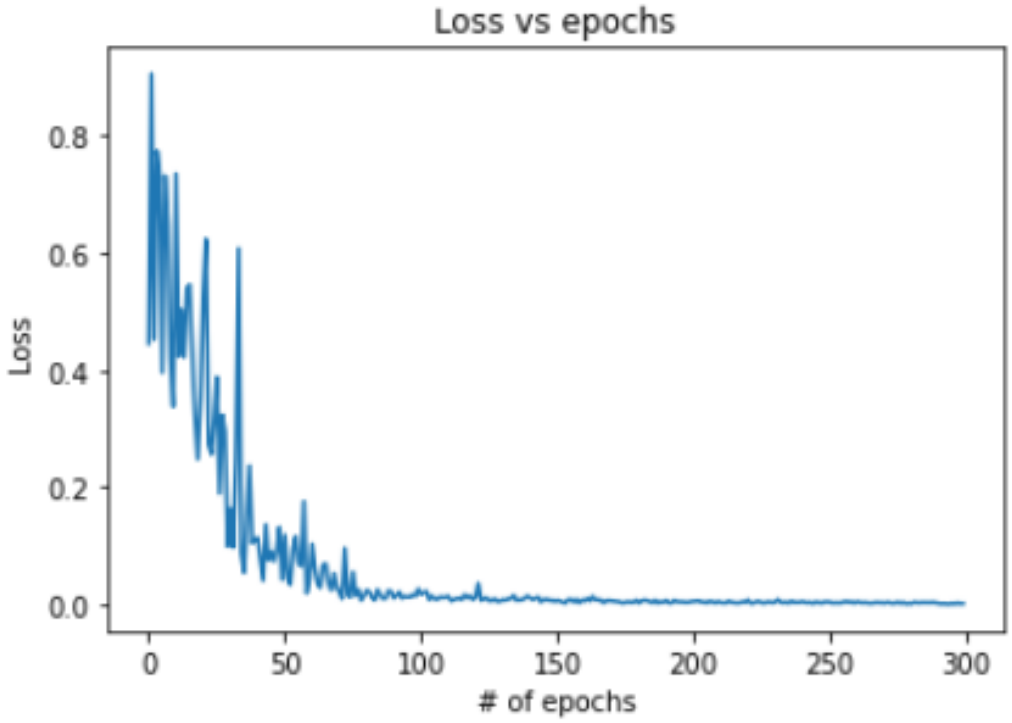
For all the problems, the CIFAR-10 data was split into training and validation sets by setting the "train" parameter to True and False, respectively, when forming each dataset. A NVIDIA RTX 3060 Ti was used for training all models.

Problem 1A:

The CIFAR-10 dataset was used to determine how well a sequential neural network (NN) will train with an image classification problem. In lecture, the CIFAR-10 dataset was reduced to classify between 2 objects rather than the 10 present in the full dataset. This allowed the NN model to perform with decent accuracy.

For this homework, all the classes present in CIFAR-10 need to be able to be identified. For Problem 1A, the NN model with one hidden layer is used to train the model. Here are the outcomes of the training performed with this model:

- Training Time: 3476 seconds or 58 minutes
- Training Loss: Reached 0.003547 after 300 epochs
- Training Accuracy: 100%
- Validation Accuracy: 53.85%
- Model Size: 6 MB
- Training Loss vs Epochs:



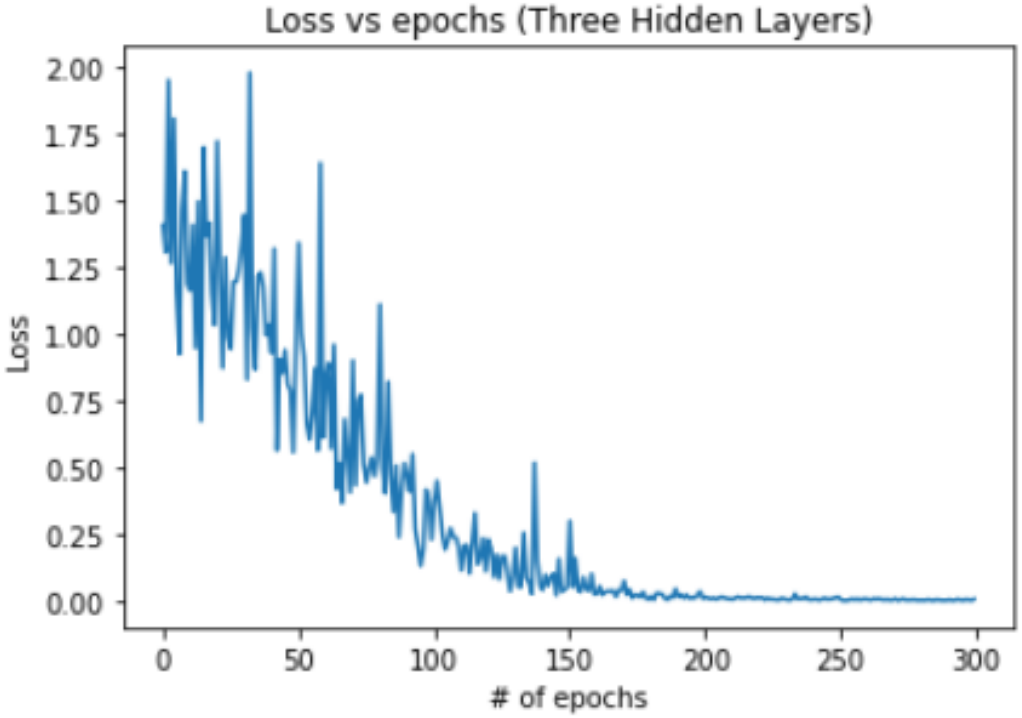
Based on the data above, it can be determined that the model trained is overfitting to the training dataset. This can be seen by the margin difference between validation accuracy and training accuracy.

Stanly Gomes
Student ID: 801118166

Problem 1B:

Due to the overfitting issue of the NN model trained above using one hidden layer, the model will be replicated with two more additional hidden layers. From previous experience, this approach allows for the model to train at a slower pace which can help with generalization. Here are the outcomes of the training performed with this model:

- Training Time: 3242 seconds or 54 minutes
- Training Loss: Reached 0.007568 after 300 epochs
- Training Accuracy: 100%
- Validation Accuracy: 52.36%
- Model Size: 14.3 MB
- Training Loss vs Epochs:



Based on the results above, the model is still overfitting to the training dataset based on the difference in accuracies. Even though the model trained at a smoother pace, the loss kept changing very sporadically. This could be improved by lowering the learning rate, but it seems the model being used is lacking an aspect needed to better classify this data.

Problem 2A:

Since the sequential NN was not showing improvements, a convolutional neural network (CNN) may yield better results. For this part of the problem, a CNN will be constructed with the following parameters:

Stanly Gomes
Student ID: 801118166

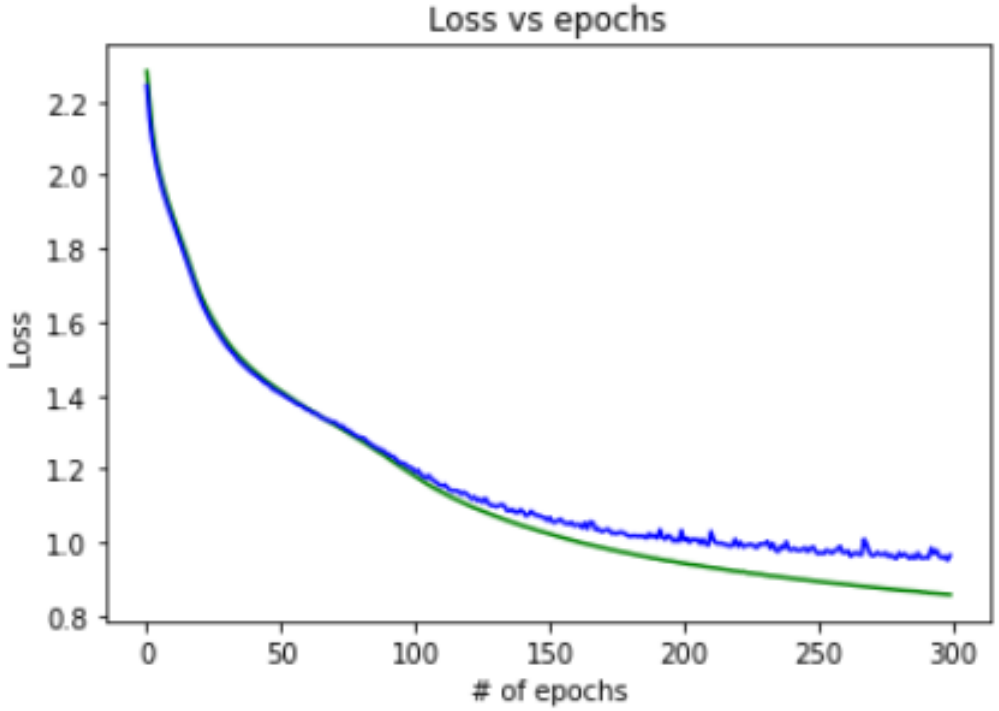
```python
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 8, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(8 * 8 * 8, 32)
        self.fc2 = nn.Linear(32, 10)

    def forward(self, x):
        out = F.max_pool2d(torch.tanh(self.conv1(x)), 2)
        out = F.max_pool2d(torch.tanh(self.conv2(out)), 2)
        out = out.view(-1, 8 * 8 * 8)
        out = torch.tanh(self.fc1(out))
        out = self.fc2(out)
        return out
```

Similar to Problem 1, the model went through a training loop to determine its performance compared to the NN model. Here are the outcomes of the training performed with this model:

- Training Time: 5524 seconds or 92 minutes
- Training Loss: Reached 0.8573 after 300 epochs
- Validation Loss: Reached 0.9641 after 300 epochs
- Training Accuracy: 69.60%
- Validation Accuracy: 66.29%
- Model Size: 74.5 KB

Stanly Gomes
Student ID: 801118166

- Training (Green)/Validation (Blue) Loss vs Epochs:



When comparing this CNN model with the NN models used in the previous problem, this model avoids overfitting to the training dataset. This can be determined by the convergence of training and validation loss seen above. A tradeoff with this was that the accuracy for both datasets were not good enough to accurately predict classes. Although, this model will perform better than the other model on new datapoints on average. Another positive to using this model was the drastic decrease in model size. Finally, this CNN model took almost twice the amount of time to train. A possible reason for this could be due to the addition of calculating validation loss every epoch using its own batch.

Problem 2B:

Since the accuracy of the first CNN was not good enough, another layer of convolution, activation function, and pooling function will be added. When performing this addition, all of the layers' inputs/outputs need to be adjusted to account for the pooling of data. Here are the parameters for this CNN model:

Stanly Gomes
Student ID: 801118166

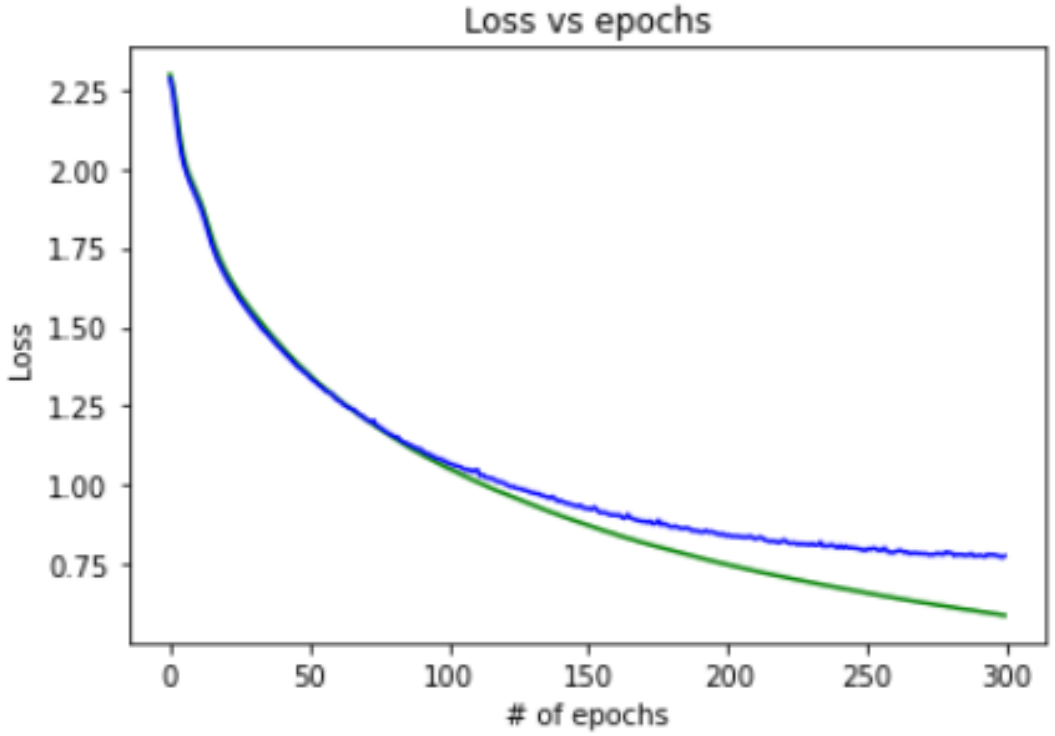
```python
class Net_3conv(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(64 * 4 * 4, 32)
        self.fc2 = nn.Linear(32, 10)

    def forward(self, x):
        out = F.max_pool2d(torch.tanh(self.conv1(x)), 2)
        out = F.max_pool2d(torch.tanh(self.conv2(out)), 2)
        out = F.max_pool2d(torch.tanh(self.conv3(out)), 2)
        out = out.view(-1, 64 * 4 * 4)
        out = torch.tanh(self.fc1(out))
        out = self.fc2(out)
        return out
```

Here are the outcomes of the training performed with this model:

- Training Time: 5670 seconds or 94 minutes
- Training Loss: Reached 0.589 after 300 epochs
- Validation Loss: Reached 0.776 after 300 epochs
- Training Accuracy: 80%
- Validation Accuracy: 73%
- Model Size: 224.9 KB

Stanly Gomes
Student ID: 801118166

- Training (Green)/Validation (Blue) Loss vs Epochs:



Similarly to before, the training for this model took almost twice the amount of time it took to train the NN models. Along with this, the model size is drastically lower than the NN model sizes, but 50% larger than the previous CNN model. An improvement has been made in the training and validation accuracy of the model compared to the original CNN model. This model will classify the 10 type of objects in the CIFAR-10 dataset much better than the other models explored in this homework. This is mostly due to the absence of overfitting in the CNN model compared to the NN model, which is very important when classifying various objects that can look drastically different in their own object type.