THE HYPERLINKS FOR THE FEATURES

- FEATURE 1: HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/16TWU9XFGE1LIAUR4UIJRWGQILPQ1RVJI?USP=SHARING
- FEATURE 2: HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/1HXADJXR9QGIG4GGWGBHAWGNKNV-FS9P5?USP=SHARING
- FEATURE 3: HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/1SUVLV6SSBVCRBE-W3IYHZBSDZXRGUBW_?USP=SHARING
- FEATURE 4: HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/1YJQVSFGZ2OGZNDIUNHUBHALRSKLTXSYN?USP=SHARING
- FEATURE 5: HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/1ZCWLJBHQVKZZ9QJMSYWITIGDLNHZNPUX?USP=SHARING

FEATURE OUTPUTS

Feature 1:

```
Files already downloaded and verified
Files already downloaded and verified
Example is of length 2
Type of first argument is <class 'torch.Tensor'>
Type of second argument is <class 'torch.Tensor'>
First argument has shape: torch.Size([64, 3, 32, 32])
Second argument has shape torch.Size([64])
Epoch 1/10: Average Loss = 1.8640, Accuracy = 36.01%
Epoch 2/10: Average Loss = 1.7672, Accuracy = 39.73%
Epoch 3/10: Average Loss = 1.7387, Accuracy = 40.82%
Epoch 4/10: Average Loss = 1.7215, Accuracy = 41.29%
Epoch 5/10: Average Loss = 1.7099, Accuracy = 41.63%
Epoch 6/10: Average Loss = 1.7013, Accuracy = 41.96%
Epoch 7/10: Average Loss = 1.6948, Accuracy = 42.46%
Epoch 8/10: Average Loss = 1.6887, Accuracy = 42.53%
Epoch 9/10: Average Loss = 1.6845, Accuracy = 42.69%
Epoch 10/10: Average Loss = 1.6804, Accuracy = 42.89%
```

Feature 2

```
Files already downloaded and verified
Files already downloaded and verified
Example is of length 2
Type of first argument is <class 'torch.Tensor'>
Type of second argument is <class 'torch.Tensor'>
First argument has shape: torch.Size([64, 3, 32, 32])
Second argument has shape torch.Size([64])
Epoch 1, Train Loss: 1.8640, Train Accuracy: 0.36
Epoch 2, Train Loss: 1.7672, Train Accuracy: 0.40
Epoch 3, Train Loss: 1.7387, Train Accuracy: 0.41
Epoch 4, Train Loss: 1.7215, Train Accuracy: 0.41
Epoch 5, Train Loss: 1.7099, Train Accuracy: 0.42
Epoch 6, Train Loss: 1.7013, Train Accuracy: 0.42
Epoch 7, Train Loss: 1.6948, Train Accuracy: 0.42
Epoch 8, Train Loss: 1.6887, Train Accuracy: 0.43
Epoch 9, Train Loss: 1.6845, Train Accuracy: 0.43
Epoch 10, Train Loss: 1.6804, Train Accuracy: 0.43
Test Loss: 0.1740, Test Accuracy: 0.40
```

Feature 3

```
Files already downloaded and verified
Files already downloaded and verified
Example is of length 2
Type of first argument is <class 'torch.Tensor'>
Type of second argument is <class 'torch.Tensor'>
First argument has shape: torch.Size([64, 3, 32, 32])
Second argument has shape torch.Size([64])
Epoch 1, Train Loss: 1.9084, Train Accuracy: 0.31
Epoch 2, Train Loss: 1.7132, Train Accuracy: 0.39
Epoch 3, Train Loss: 1.6430, Train Accuracy: 0.41
Epoch 4, Train Loss: 1.5961, Train Accuracy: 0.43
Epoch 5, Train Loss: 1.5617, Train Accuracy: 0.44
Epoch 6, Train Loss: 1.5315, Train Accuracy: 0.45
Epoch 7, Train Loss: 1.5051, Train Accuracy: 0.46
Epoch 8, Train Loss: 1.4858, Train Accuracy: 0.47
Epoch 9, Train Loss: 1.4648, Train Accuracy: 0.48
Epoch 10, Train Loss: 1.4501, Train Accuracy: 0.48
Train Loss: 0.0015, Train Accuracy: 0.61
Test Loss: 0.1322, Test Accuracy: 0.53
```

Insights :

Train Loss: 0.0015

Train Accuracy: 61%

Test Loss: 0.1322

Test Accuracy: 50%

I concluded that my model is an overfitting one since the training is more accurate than testing. I cannot blame the data set since it's already huge but I am thinking I could apply regularization.

Feature 4

```
Files already downloaded and verified
Files already downloaded and verified
Example is of length 2
Type of first argument is <class 'torch.Tensor'>
Type of second argument is <class 'torch.Tensor'>
First argument has shape: torch.Size([64, 3, 32, 32])
Second argument has shape torch.Size([64])
Epoch 1, Train Loss: 1.7431, Train Accuracy: 0.38
Epoch 2, Train Loss: 1.4545, Train Accuracy: 0.49
Epoch 3, Train Loss: 1.3240, Train Accuracy: 0.53
Epoch 4, Train Loss: 1.2212, Train Accuracy: 0.57
Epoch 5, Train Loss: 1.1418, Train Accuracy: 0.60
Epoch 6, Train Loss: 1.0677, Train Accuracy: 0.63
Epoch 7, Train Loss: 1.0019, Train Accuracy: 0.65
Epoch 8, Train Loss: 0.9472, Train Accuracy: 0.67
Epoch 9, Train Loss: 0.8976, Train Accuracy: 0.69
Epoch 10, Train Loss: 0.8541, Train Accuracy: 0.70
Train Loss: 0.0011, Train Accuracy: 0.64
Test Loss: 0.1092, Test Accuracy: 0.61
```

Insights:

Train Loss: 0.0011

Train Accuracy: 64%

Test Loss: 0.1092

Test Accuracy: 61%

I concluded that my CNN model is slightly overfitting since the training output is just 3% more accurate than the test one. The performance has definitely improved a lot when compared to MLP

Feature 5

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|██████████| 170498071/170498071 [00:03<00:00, 43493492.39it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
Example is of length 2
Type of first argument is <class 'torch.Tensor'>
Type of second argument is <class 'torch.Tensor'>
First argument has shape: torch.Size([64, 3, 32, 32])
Second argument has shape torch.Size([64])
Epoch 1, Train Loss: 1.7879, Train Accuracy: 0.37
Epoch 2, Train Loss: 1.5355, Train Accuracy: 0.46
Epoch 3, Train Loss: 1.4273, Train Accuracy: 0.49
Epoch 4, Train Loss: 1.3470, Train Accuracy: 0.52
Epoch 5, Train Loss: 1.2996, Train Accuracy: 0.54
Epoch 6, Train Loss: 1.2534, Train Accuracy: 0.56
Epoch 7, Train Loss: 1.2076, Train Accuracy: 0.58
Epoch 8, Train Loss: 1.1683, Train Accuracy: 0.59
Epoch 9, Train Loss: 1.1372, Train Accuracy: 0.60
Epoch 10, Train Loss: 1.1125, Train Accuracy: 0.61
Train Loss: 0.0016, Train Accuracy: 0.56
Test Loss: 0.1185, Test Accuracy: 0.56
```

Insights:

Train Loss: 0.0016, Train Accuracy: 0.56

Test Loss: 0.1185, Test Accuracy: 0.56

There is a reduction in training accuracy which is understandable since I augmented the data but I effectively reduced overfitting which makes me happy 😊