1. **What do the 0.5's represent?**
   The 0.5s in the Normalize function represent the mean and standard deviation values for each RGB channel of the images.

2. **What do the arguments in the conv2d represent?**
   - **in_channels:** Number of input channels (e.g., 3 for RGB images).
   - **out_channels:** Number of filters to apply, determining the output channels.
   - **kernel_size:** Size of each filter (e.g., 5 for a 5x5 filter).
   - **stride:** Steps the filter takes when sliding over the input (default is 1).
   - **padding:** Adds border pixels around the input to control output size (default is 0).

3. **What do the arguments in the MaxPool represent?**
   - **kernel_size:** The size of the window to take a max over (e.g., 2 for a 2x2 window).
   - **stride:** The number of pixels the window moves over the input (default is the same as kernel_size if not specified).
4. **What do the arguments in the Linear represent?**
   - **in_features:** The number of input features (nodes) to the layer.
   - **out_features:** The number of output features (nodes) the layer will produce
5. **What is the flatten layer doing?**
   The flatten layer reshapes the multi-dimensional tensor output from previous layers into a 1D vector. This is essential for transitioning from the convolutional layers (which output 2D feature maps) to the fully connected (linear) layers that expect a 1D input.
6. **Create a separate script file to load in the saved model and do the above prediction.**
   See train10.py and predict10.py

```
gbaks@gb_laptop MINGW64 ~/Downloads/cis
nt11-cnn (main)
$ python train10.py
Files already downloaded and verified
Files already downloaded and verified
[1,  2000] loss: 2.177
[1,  4000] loss: 1.764
[2,  2000] loss: 1.516
[2,  4000] loss: 1.438

gbaks@gb_laptop MINGW64 ~/Downloads/cis
nt11-cnn (main)
$ python predict10.py
Files already downloaded and verified
Files already downloaded and verified
Accuracy of the network on the 10000 te
Accuracy for class: plane is 58.9 %
Accuracy for class: car   is 76.2 %
Accuracy for class: bird  is 32.2 %
Accuracy for class: cat   is 25.6 %
Accuracy for class: deer  is 38.6 %
Accuracy for class: dog   is 48.5 %
Accuracy for class: frog  is 74.0 %
Accuracy for class: horse is 56.9 %
Accuracy for class: ship  is 56.8 %
Accuracy for class: truck is 56.6 %
```

7. **Modify the above code to only learn 5 classes**
   See train5.py and predict5.py

```
gbaks@gb_laptop MINGW64 ~/Downloads/cis3
nt11-cnn (main)
$ python train5.py
Files already downloaded and verified
Files already downloaded and verified
[1,  2000] loss: 1.537
[1,  4000] loss: 1.184
[1,  6000] loss: 1.022
[2,  2000] loss: 0.935
[2,  4000] loss: 0.915
[2,  6000] loss: 0.878
Finished Training

gbaks@gb_laptop MINGW64 ~/Downloads/cis3
nt11-cnn (main)
$ python predict5.py
Files already downloaded and verified
Files already downloaded and verified
Accuracy on test images: 63.80 %
Accuracy for class plane: 84.5 %
Accuracy for class car  : 86.4 %
Accuracy for class bird : 59.3 %
Accuracy for class cat  : 52.0 %
Accuracy for class dog  : 36.8 %
```

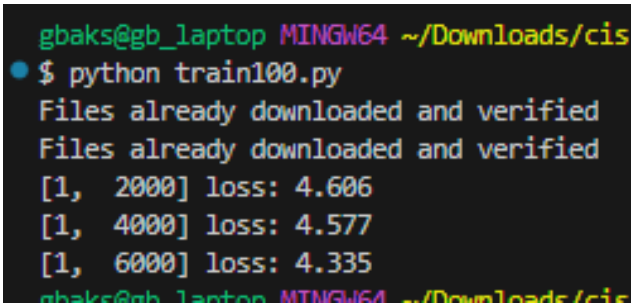
8. **Modify the above code to only learn 2 classes**
   See train2.py and predict2.py

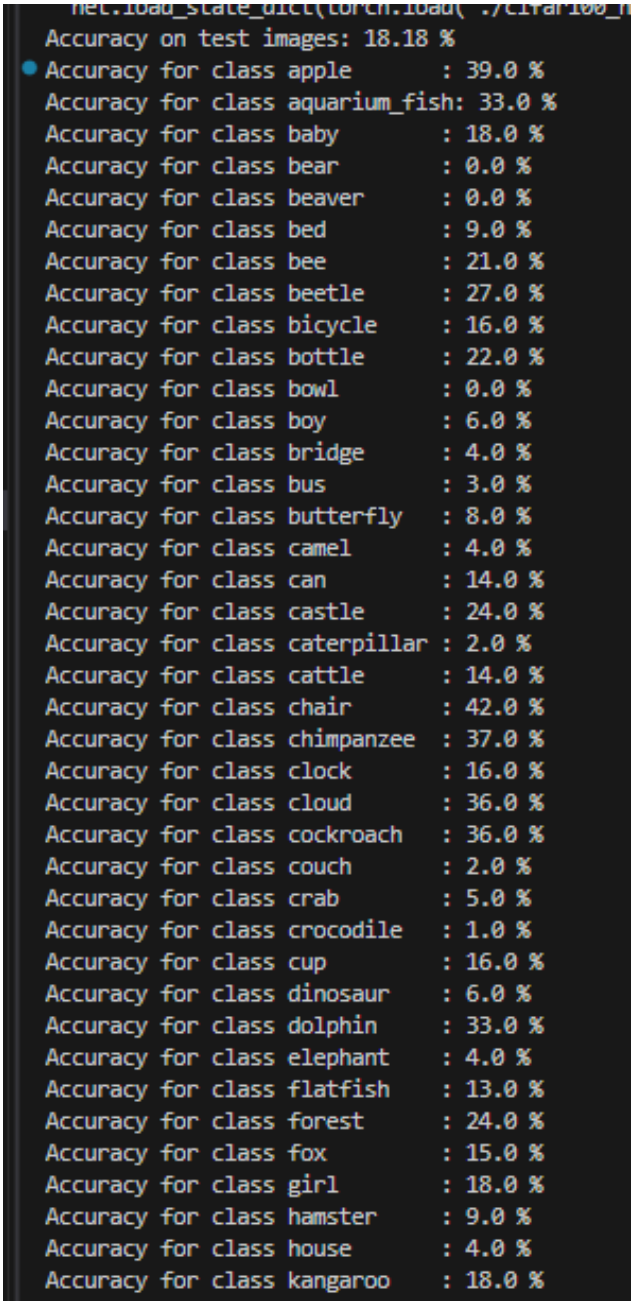
```
gbaks@gb_laptop MINGW64 ~/Downloads/cis3
nt11-cnn (main)
$ python train2.py
Files already downloaded and verified
Files already downloaded and verified
[1,  2000] loss: 0.564
[2,  2000] loss: 0.321
Finished Training

gbaks@gb_laptop MINGW64 ~/Downloads/cis3
nt11-cnn (main)
$ python predict2.py
Files already downloaded and verified
Files already downloaded and verified
Accuracy on test images: 89.25 %
Accuracy for class plane: 92.7 %
Accuracy for class car  : 85.8 %
```

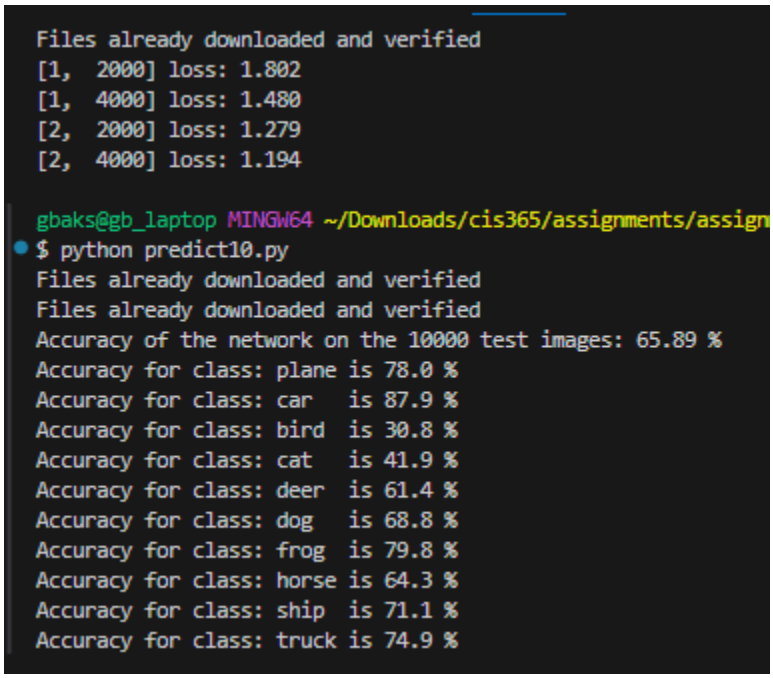
9. **Repeat the above exercise but using the CIFAR-100 dataset located here:**

   See train100.py and predict100.py





10. **Can you improve the performance or your neural network by changing the layers/parameters of the model?**
    I messed around with the configuration of the cnn and was testing it with the 10 cifar10 classes and was able to get it up to 65.89%. Generally, birds and cats did the worst, bringing down the average.



11. **Which group was able to achieve the best performance?**
    Cars did the best, getting 87.9% when classifying 10 classes, probably because of its distinct features.