Dinh Thien Tu Tran
CSCI 611
Professor Bo Shen

Task 1: CNN Design and Training

I take the skeleton code from Canvas and work my way from there.

My first action was to look at *build_cnn.ipynb* to build the CIFAR-10 dataset which consists of 60000 32x32 colour images in 10 classes, including 50000 training images and 10000 testing images. More details for the CIFAR-10 database here.

I set batch size = 64 for the model training.
The convolutional network architecture consists of three convolutional layers followed by fully connected layers for classification.

```
self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stride=1, padding=1)
self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1)
self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1)

# max pooling layer
self.pool = nn.MaxPool2d(2, 2) # reduce h,w by half
```

```
x = self.pool(F.relu(self.conv1(x)))
x = self.pool(F.relu(self.conv2(x)))
x = F.relu(self.conv3(x))
```

Layer 1: 3 input channels to 32 output channels
Layer 2: 32 to 64 channels
Layer 3: 64 to 128 channels
For the first 2 layers, they are followed by ReLU activation and max pooling.
The 3rd layer needs ReLU, no max pooling.

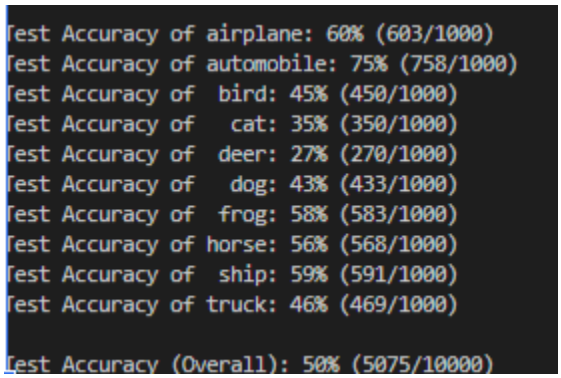
After going through 3 convolutional layers, the feature maps are flattened and passed into fully connected layers.
Dropout was applied before and between fully connected layers to reduce overfitting.
No additional data augmentation was used.

The loss function I used is CrossEntropyLoss, appropriate for multi-class classification problems.
The optimizer I initially used is SGD with learning rate 0.01, but switched to Adam with learning rate 0.01 later for better performance.

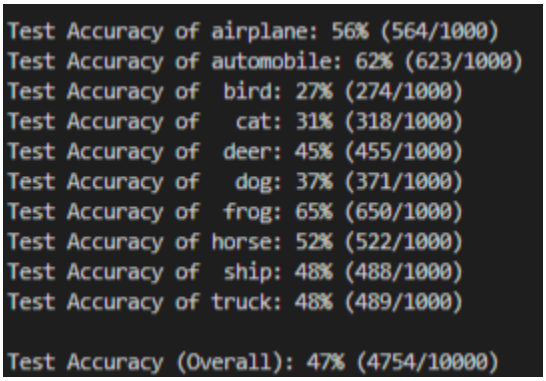
Training model with epochs = 5

Result comparison between SGD and Adam:

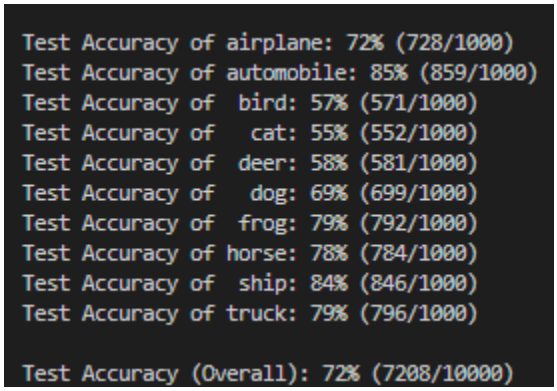
SGD (lr = 0.01)



Adam (0.01)



So I did some research and it appears that setting lr = 0.01 is a very uncommon value for Adam optimizer so I switched to 0.001 and got a better result. I also tested with different learning rate values but 0.001 seems to yield the overall better result.



As for the number of epochs, testing with multiple values let me know that the optimal number is around 5 to 7. Although setting a higher epoch number does result in higher test accuracy, but our validation value will also increase along the way, thus increasing the chance of model overfitting.

Example of epoch = 10:

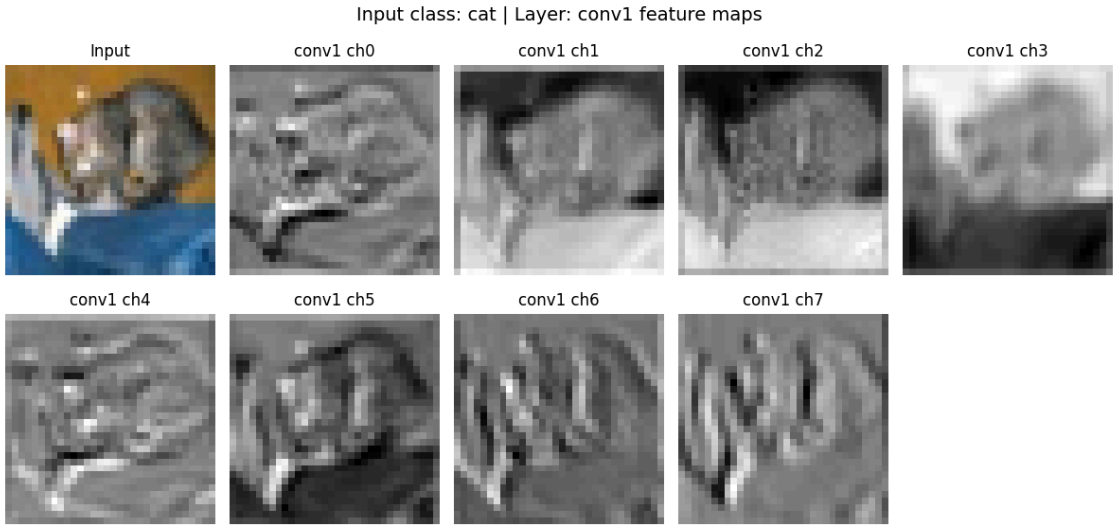
```
Epoch: 2        Training Loss: 1.051954         Validation Loss: 1.035916
Validation loss decreased (1.128921 --> 1.035916).  Saving model ...
Epoch: 3        Training Loss: 0.865708         Validation Loss: 0.821386
Validation loss decreased (1.035916 --> 0.821386).  Saving model ...
Epoch: 4        Training Loss: 0.736619         Validation Loss: 0.771468
Validation loss decreased (0.821386 --> 0.771468).  Saving model ...
Epoch: 5        Training Loss: 0.629840         Validation Loss: 0.744619
Validation loss decreased (0.771468 --> 0.744619).  Saving model ...
Epoch: 6        Training Loss: 0.546654         Validation Loss: 0.730709
Validation loss decreased (0.744619 --> 0.730709).  Saving model ...
Epoch: 7        Training Loss: 0.474510         Validation Loss: 0.737737
Epoch: 8        Training Loss: 0.402902         Validation Loss: 0.736225
Epoch: 9        Training Loss: 0.341174         Validation Loss: 0.785104
Epoch: 10       Training Loss: 0.293113         Validation Loss: 0.828971
Finished Training
Test Loss: 0.749607

Test Accuracy of airplane: 84% (848/1000)
Test Accuracy of automobile: 86% (865/1000)
Test Accuracy of  bird: 52% (528/1000)
Test Accuracy of   cat: 51% (513/1000)
Test Accuracy of  deer: 80% (807/1000)
Test Accuracy of   dog: 67% (671/1000)
Test Accuracy of  frog: 78% (785/1000)
Test Accuracy of horse: 80% (806/1000)
Test Accuracy of  ship: 85% (851/1000)
Test Accuracy of truck: 80% (802/1000)

Test Accuracy (Overall): 74% (7476/10000)
```

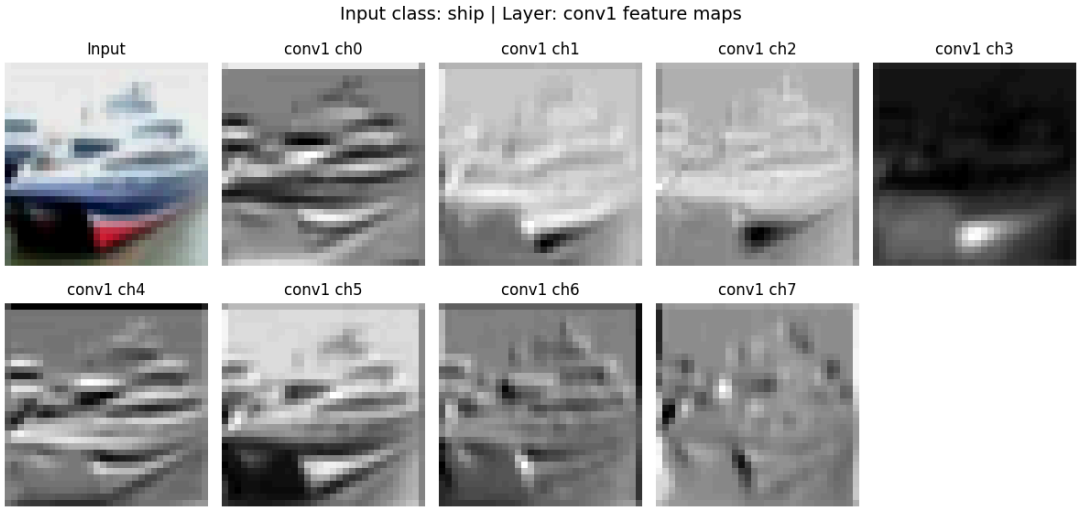
Task 2: Feature Map Visualization

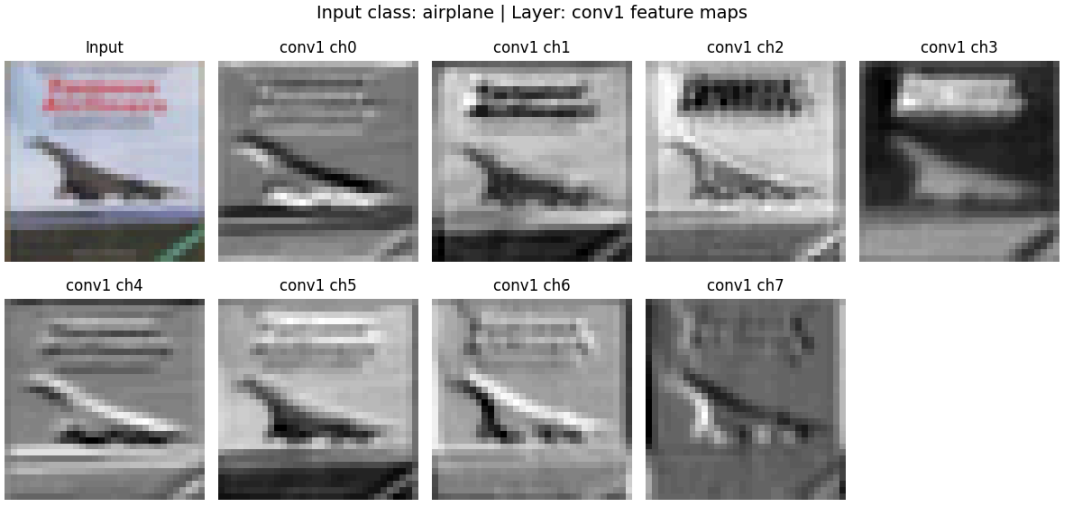
Part A: Feature Maps from the First Convolutional Layer

1. Select 3 test images from different classes.
2. For each image:
   Pass it through the trained CNN.
   Extract feature maps from the first convolutional layer.
3. Visualize at least 8 feature maps per image

Codes for this part are updated on Github with detailed explanations.


Input class: cat | Layer: conv1 feature maps
Input
conv1 ch0
conv1 ch1
conv1 ch2
conv1 ch3
conv1 ch4
conv1 ch5
conv1 ch6
conv1 ch7


Input class: ship | Layer: conv1 feature maps
Input
conv1 ch0
conv1 ch1
conv1 ch2
conv1 ch3
conv1 ch4
conv1 ch5
conv1 ch6
conv1 ch7


Input class: airplane | Layer: conv1 feature maps
Input
conv1 ch0
conv1 ch1
conv1 ch2
conv1 ch3
conv1 ch4
conv1 ch5
conv1 ch6
conv1 ch7

The feature maps from the first convolutional layer reveal that the CNN learns low-level visual features such as edges, gradients, and simple textures. For the cat image, several channels strongly activate along the boundaries of the cat's head and ears, but stop making sense and lose edges around channel 6 and 7. For the airplane image, several channels strongly activate along diagonal edges corresponding to the wing and body, suggesting that the filters detect edges. As of the ship image, the channels emphasize horizontal structures such as the deck edges and it reacts strongly to the red stripe at the bottom of the ship as we can clearly notice it is reflected in every channel.

Different feature maps respond differently to the same input, with some highlighting boundaries while some emphasize background regions or texture patterns. In all, we can understand that the first layer of convolution works as a collection that extracts fundamental visual features from the inputs.

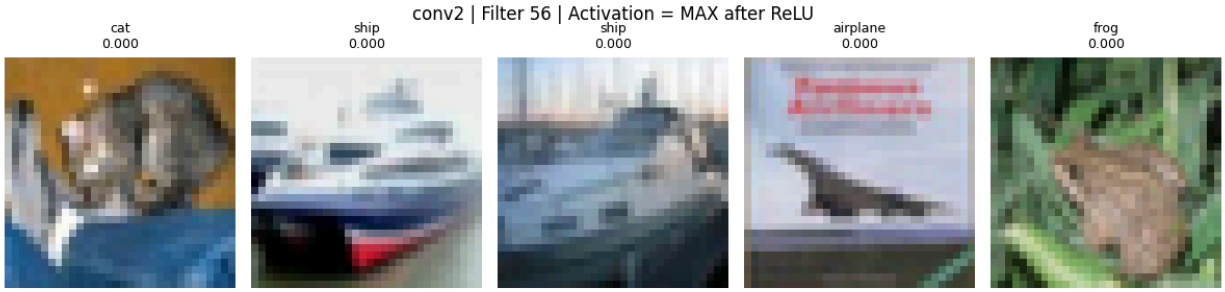
Part B: Maximally Activating Images for Selected Filters

For this part, I choose to define activation of a filter for an image as the **maximum** value in the feature map.

Common visual patterns:

Strong edges, centered objects, high difference in contrast between object and background

It seems to have a strong reaction to automobile class.



Common visual patterns:

All activation scores are 0. What this means is that after ReLU activation, the filter produced no positive activations. It does not appear to respond strongly to general or class-specific patterns.

Common visual patterns:
A very specific set of colors appear: blue, smooth background, bright contrast between object and background. This filter is not class specific, it responds strongly to blue color dominant images.

Overall, the results demonstrate that mid-level convolutional filters capture a mixture of structural and color features, some appear to be highly specialized while others may be inactive.