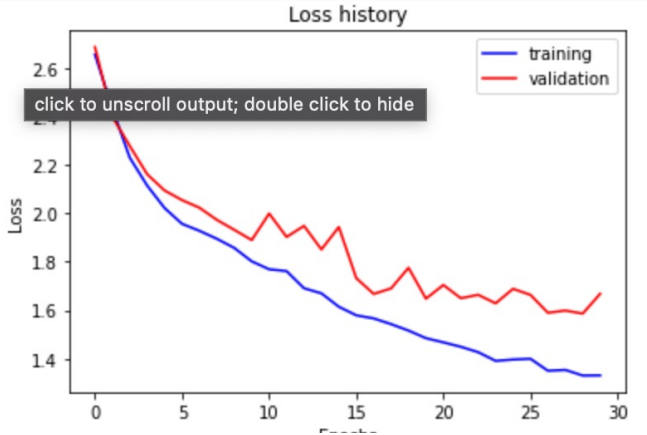
CS 6476 Project 5

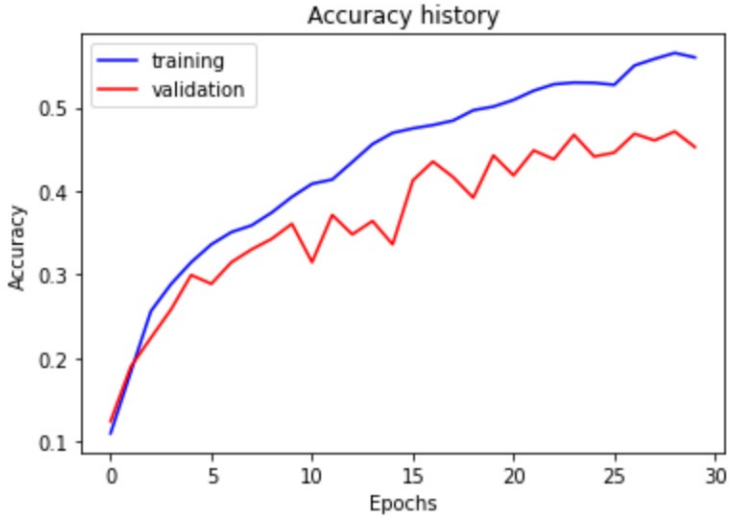
Haoran Wang
Haoran.wang@gatech.edu
Hwang827
903543952

Part 1: SimpleNet

[Insert loss plot for SimpleNet here]



[Insert accuracy plot for SimpleNet here]



Final training accuracy: 0.5601340033500838

Final validation accuracy: 0.45266666666666666

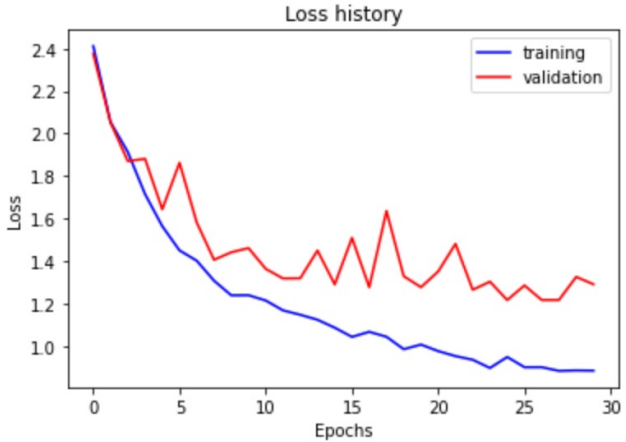
Part 2: SimpleNetFinal

Add each of the following (keeping the changes as you move to the next row):

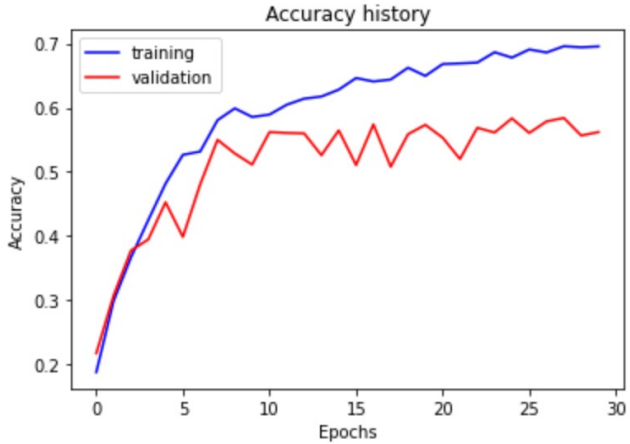
	Training accuracy	Validation accuracy
SimpleNet	0.5601340033500838	0.45266666666666666
+ Jittering	0.5045226130653266	0.44533333333333336
+ Zero-centering & variance-normalization	0.7128978224455611	0.5493333333333333
+ Dropout regularization	0.6418760469011725	0.5586666666666666
+ Making network "deep"	0.6579564489112227	0.5346666666666666
+ Batch normalization	0.6954773869346733	0.562

Part 2: SimpleNetFinal

[Insert loss plot for SimpleNetFinal here]



[Insert accuracy plot for SimpleNetFinal here]



Final training accuracy: 0.6954773869346733

Final validation accuracy: 0.562

Part 2: SimpleNetFinal

[Name 10 different possible transformations for data augmentation.]
Position augmentation:

1. Scaling
2. Cropping
3. Flipping
4. Padding
5. Rotation, Translation
6. Affine transformation

Color augmentation:

1. Brightness
2. Contrast
3. Saturation
4. Hue

[What is the desired variance after each layer? Why would that be helpful?]
The variance are RandomHorizontalFlip, ColorJitter, Normalize. By flipping the image horizontally, and jittering the color, we can increase our amount of training data during the learning process without changing the label of the image. By normalizing with mean and standard deviation of the images, standardizing the greyscale image and centering at zero, so that all the training are at the same numerical scale.

Part 2: SimpleNetFinal

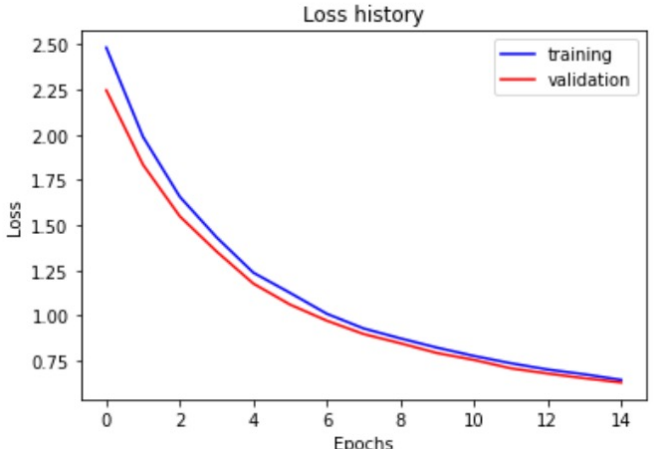
[What distribution is dropout usually sampled from?]
Gaussian distribution, since it randomly drops out.

[How many parameters does your base SimpleNet model have? How many parameters does your SimpleNetFinal model have?]
SimpleNet: 5,280
SimpleNetFinal: 32,235

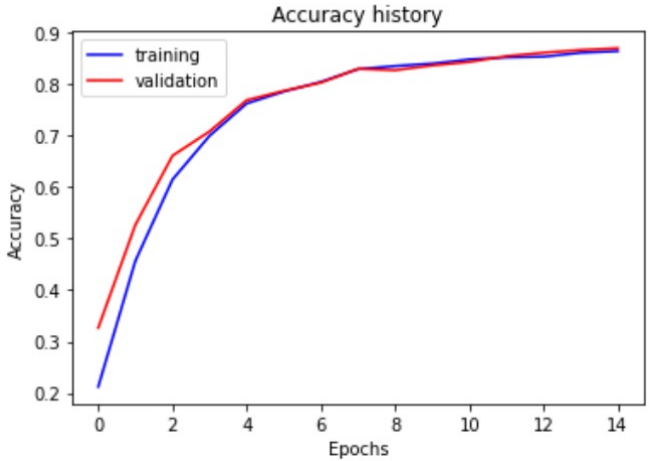
[What is the effect of batch norm after a conv layer with a bias?]
It standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

Part 3: ResNet

[Insert loss plot here]



[Insert accuracy plot here]

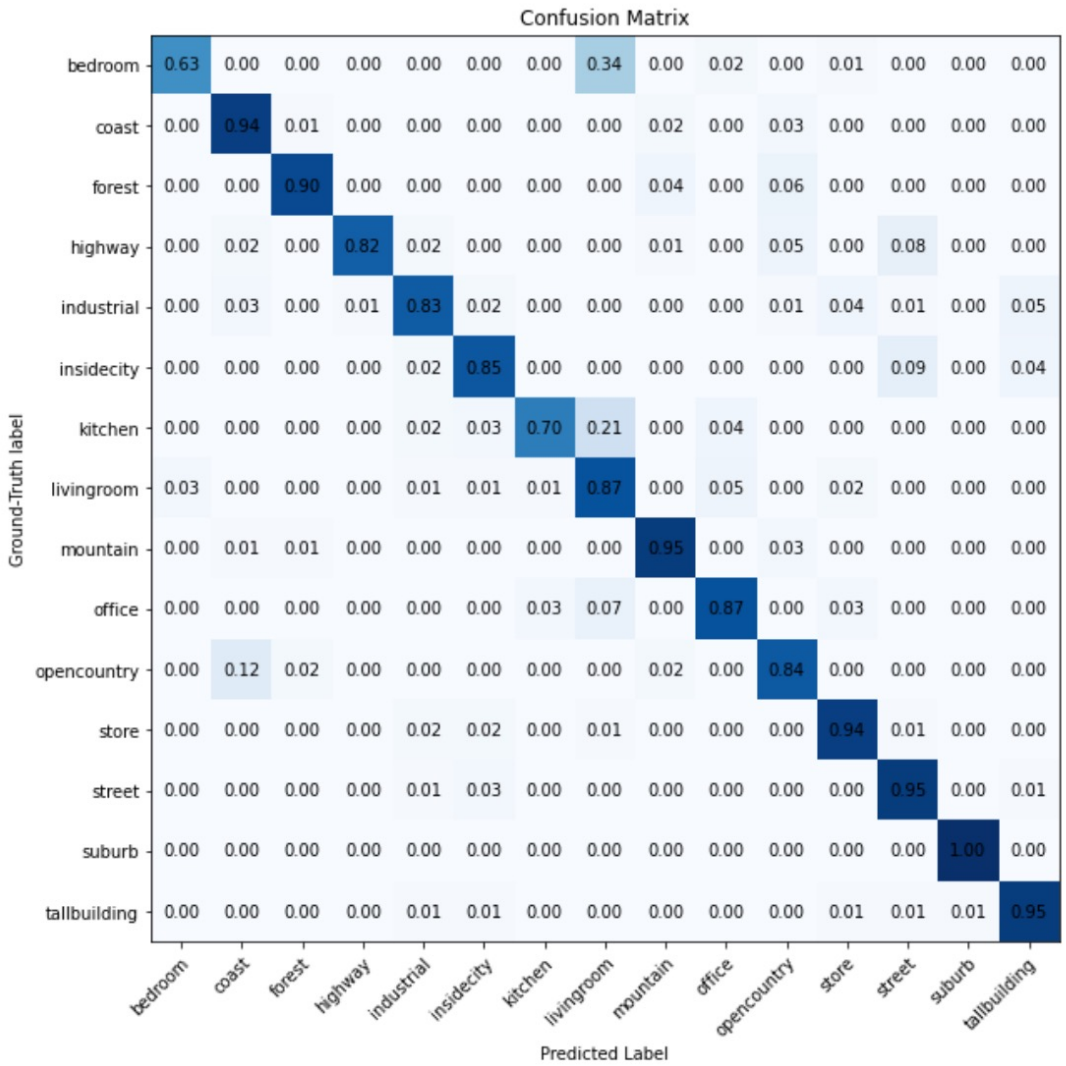


Final training accuracy: 0.8636515912897822

Final validation accuracy: 0.8693333333333333

Part 3: ResNet

[Insert visualization of confusion matrix obtained from your final ResNet model.]



Part 3: ResNet

[Insert visualizations of 3 misclassified images from the most misclassified class according to your confusion matrix. Explain why this may have occurred.]
Because all of these bedrooms has quite a few furniture like table or coach, which are obvious in living room pictures, and these bedrooms do not have an very obvious bed, which confused the ResNet to classify them as a living room.

Part 3: ResNet

[What does fine-tuning a network mean?]
Fine-tuning takes a model that has already been trained for a particular task and then fine-tuning or tweaking it to make it perform a second similar task.

[Why do we want to “freeze” the conv layers and some of the linear layers from a pre-trained ResNet? Why can we do this?]
Freezing a layer means the layer doesn’t need any modification to the data contained in them, henceforth. The weights for these layers don’t update when we train the new model on the new data for the new task. Freezing some layers of the pre-trained ResNet is to cut down on the computational time for training while losing not much on the accuracy side.

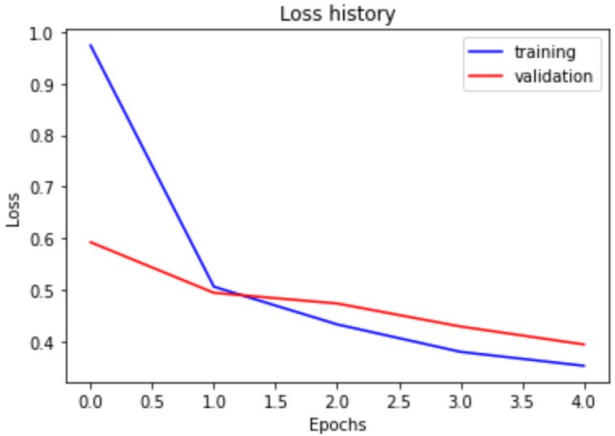
Extra credit (optional)

I also trained an customized AlexNet from the pre-trainined AlexNet model, freezing the convolutional layers and modified the last Linear layer in the fully connected layer to fit our data. Architecture shown on the right. I reached an accuracy of more than 85% after 5 training epoch. Plots refer to next page.

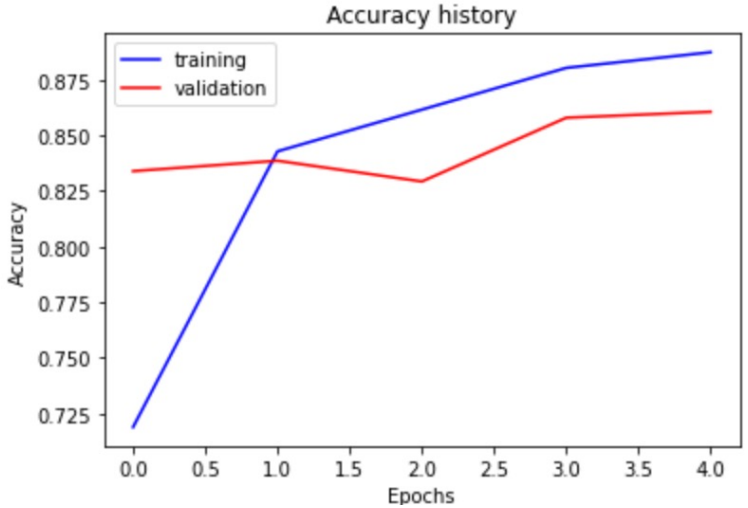
```
MyAlexNet(
  (conv_layers): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
      (1): ReLU(inplace=True)
      (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
      (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
      (4): ReLU(inplace=True)
      (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
      (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (7): ReLU(inplace=True)
      (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (9): ReLU(inplace=True)
      (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (1): AdaptiveAvgPool2d(output_size=(6, 6))
  )
  (fc_layers): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=15, bias=True)
  )
  (loss_criterion): CrossEntropyLoss()
)
```

Extra credit (optional)

[Insert loss plot for MyAlexNet here]



[Insert accuracy plot here]



Final training accuracy: 0.8874371859296483

Final validation accuracy: 0.8606666666666667

Extra credit (optional)

- [Insert visualization of confusion matrix obtained from your final MyAlexNet model.]
- AlexNet trains much faster than ResNet18, with the same learning rate = 0.004, weight decay = 0.002 and 5 epoch. While ResNet take about 5 to 10 minutes training on colab, AlexNet take only about 1 minute to train. They reach similar training and validation accuracies after the above learning. However, MyAlexNet model takes 223 MB, which is much bigger than MyRestNet model 44.8Mb.
- I didn't include the trained MyAlexNet model in the submission, since it is too large. For code of MyAlexNet, please refer to my_alexnet.py.