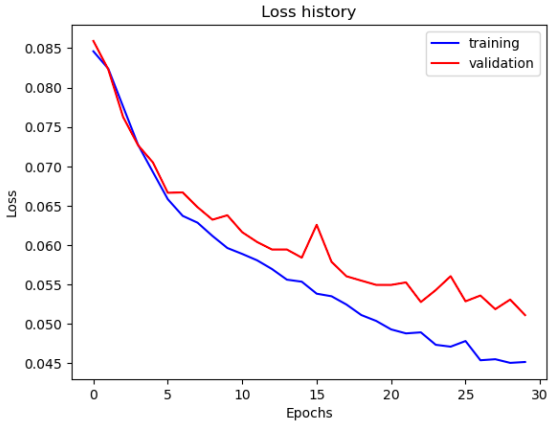
CS 4476 Project 4

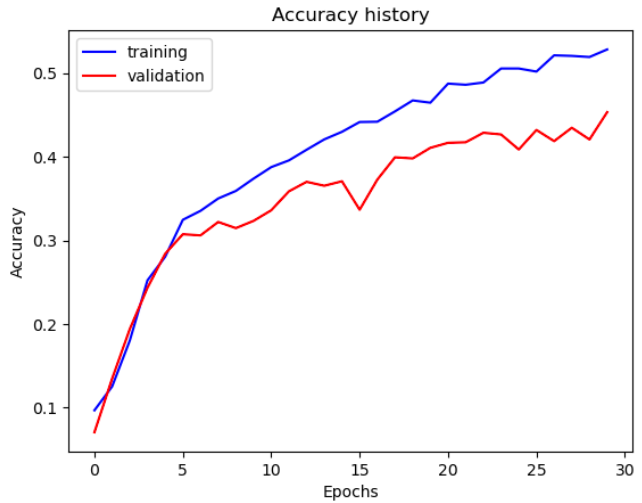
[Seohee Yoon]
[syoon333@gatech.edu]
[syoon333]
[903763900]

Part 1: SimpleNet

[Insert loss plot for SimpleNet here]



[Insert accuracy plot for SimpleNet here]



Final training accuracy: 0.5283082077051926

Final validation accuracy: 0.4533333333333333

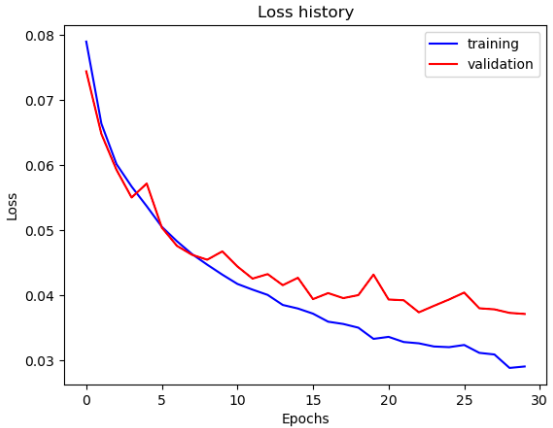
Part 2: SimpleNetFinal

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

	Training accuracy	Validation accuracy
SimpleNet	0.528308207	0.453333333
+ Jittering	0.4877721943	0.4426666667
+ Zero-centering & variance-normalization	0.5953098827	0.4986666667
+ Dropout regularization	0.6301507537	0.5106666667
+ Making network "deep"	0.6539363484	0.5646666667
+ Batch normalization	0.6921273032	0.589333333

Part 2: SimpleNetFinal

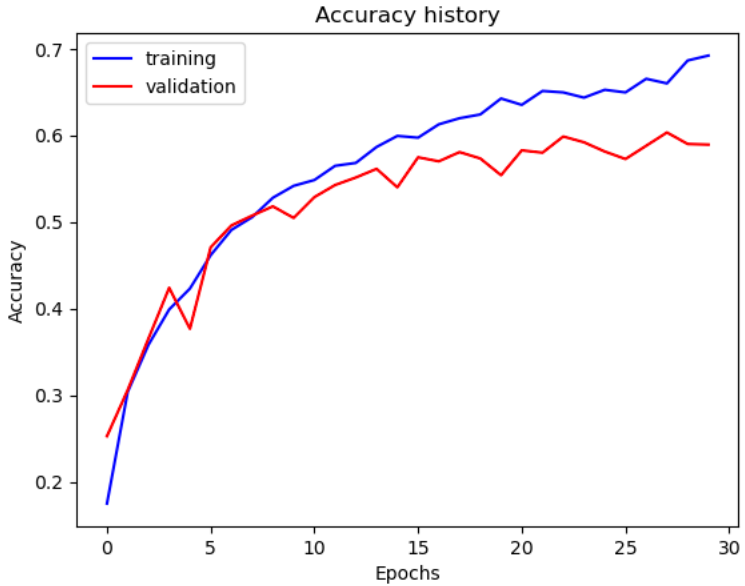
[Insert loss plot for SimpleNetFinal here]



Final training accuracy: 0.6921273031825795

Final validation accuracy: 0.5893333333333334

[Insert accuracy plot for SimpleNetFinal here]



Part 2: SimpleNetFinal

[Name 10 different possible transformations for data augmentation.]

1. Padding
2. Cropping
3. Color Jitter
4. Rotation
5. Translation
6. Flipping
7. Brightness
8. Rescaling
9. Zooming
10. Shearing

[What is the desired variance after each layer? Why would that be helpful?]

The desired variance after each layer is the same variance throughout layers. This can ensure learning process stable, and prevent issues such as vanishing.

Part 2: SimpleNetFinal

[What distribution is dropout usually sampled from?]
Dropout is usually sampled from a Bernoulli distribution.

[How many parameters does your base SimpleNet model have? How many parameters does your SimpleNetFinal model have?]

SimpleNet = 8 parameters

SimpleNetFinal = 14 parameters

[What is the effect of batch norm after a conv layer with a bias?]

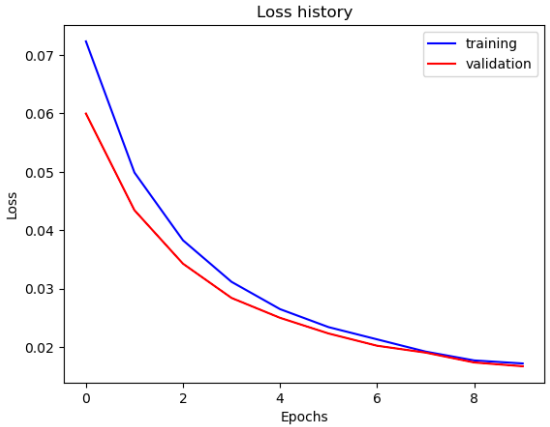
The effect can include normalization the activations of a layer by subtracting the mean and dividing by std. If a conv layer has a bias, this can affect the computation of mean and variance during the normalization process.

```
<class 'torch.nn.parameter.Parameter'> torch.Size([10, 1, 5, 5])
<class 'torch.nn.parameter.Parameter'> torch.Size([10])
<class 'torch.nn.parameter.Parameter'> torch.Size([20, 10, 5, 5])
<class 'torch.nn.parameter.Parameter'> torch.Size([20])
<class 'torch.nn.parameter.Parameter'> torch.Size([100, 500])
<class 'torch.nn.parameter.Parameter'> torch.Size([100])
<class 'torch.nn.parameter.Parameter'> torch.Size([15, 100])
<class 'torch.nn.parameter.Parameter'> torch.Size([15])
```

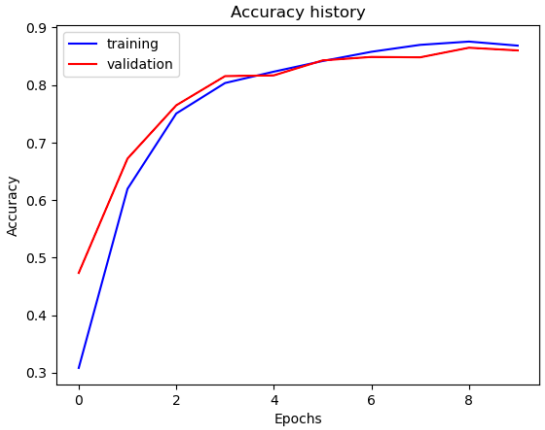
```
<class 'torch.nn.parameter.Parameter'> torch.Size([10, 1, 5, 5])
<class 'torch.nn.parameter.Parameter'> torch.Size([10])
<class 'torch.nn.parameter.Parameter'> torch.Size([10])
<class 'torch.nn.parameter.Parameter'> torch.Size([10])
<class 'torch.nn.parameter.Parameter'> torch.Size([20, 10, 5, 5])
<class 'torch.nn.parameter.Parameter'> torch.Size([20])
<class 'torch.nn.parameter.Parameter'> torch.Size([20])
<class 'torch.nn.parameter.Parameter'> torch.Size([20])
<class 'torch.nn.parameter.Parameter'> torch.Size([100, 20, 5, 5])
<class 'torch.nn.parameter.Parameter'> torch.Size([100])
<class 'torch.nn.parameter.Parameter'> torch.Size([100, 900])
<class 'torch.nn.parameter.Parameter'> torch.Size([100])
<class 'torch.nn.parameter.Parameter'> torch.Size([15, 100])
<class 'torch.nn.parameter.Parameter'> torch.Size([15])
```

Part 3: ResNet

[Insert loss plot here]



[Insert accuracy plot here]

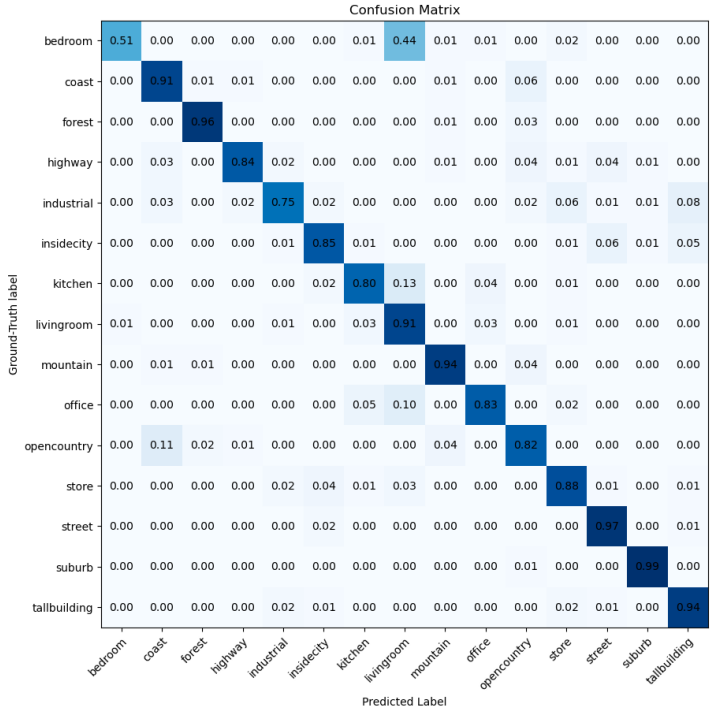


Final training accuracy: 0.8683417085427135

Final validation accuracy: 0.86

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.]



According to confusion matrix, our model classified image that was actually bedroom as living room. I think these images that I inserted have similar features to living room such as furniture or windows. This can make our model confuse to differentiate between living room and bedroom.

Part 3: ResNet

[What does fine-tuning a network mean?]

Fine-tuning a network mean the process of using the weights of an already trained network as the starting values for training a new new network.
We can improve the efficiency and performance of our new model that we want to train by using pre-trained models to utilize the knowledge captured by the pre-trained models.

[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?]

Because we want to use the pre-trained model as fixed feature extractor by using the knowledge captured in freezed layers for our goal without changing the learned features.
We can do freeze the conv layers due to the nature of transfer learning and the architecture of neural networks.

Part 4: Multi-label Scene Attributes

[Insert loss plot here]

[Insert accuracy plot here]

Final training accuracy:

Final validation accuracy:

Part 4: Multi-label Scene Attributes

[Insert visualization of accuracy table obtained from your final MultilabelResNet model.]

Part 4: Multi-label Scene Attributes

[List 3 changes that you made in the network compared to the one in part 3.]

[Is the loss function of the ResNet model from part 3 appropriate for this problem? Why or why not?]

Part 4: Multi-label Scene Attributes

[Explain a problem that one needs to be wary of with multilabel classification. HINT: consider the purpose of visualizing your results with the accuracy table. You might want to do some data exploration here.]