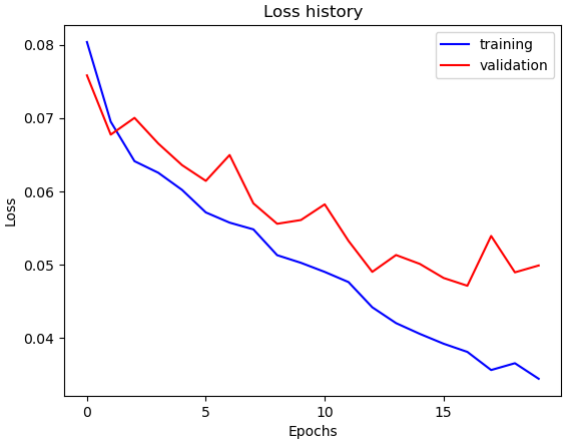
CS 4476/6476 Project 4

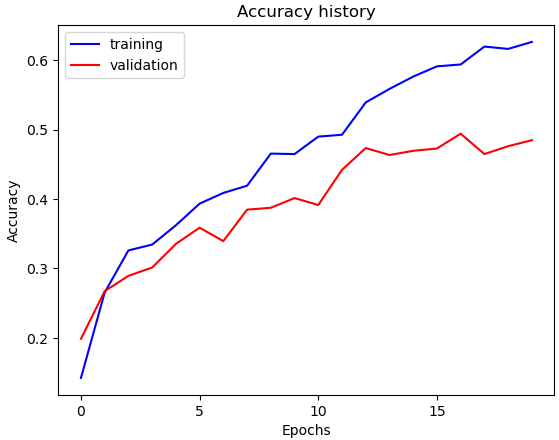
Shuvo Newaz
shuvo.newaz@gatech.edu
snewaz3
903614132

Part 1: SimpleNet

[Insert loss plot for SimpleNet here]



[Insert accuracy plot for SimpleNet here]



Final training accuracy: 62.6%

Final validation accuracy: 48.4%

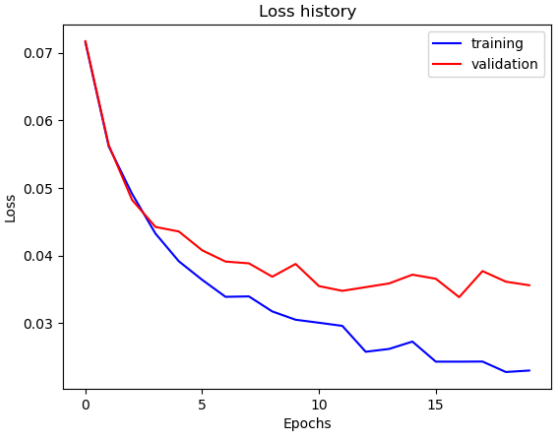
Part 2: SimpleNetFinal

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

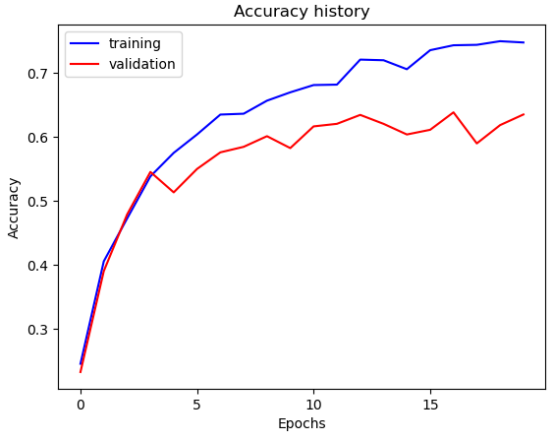
	Training accuracy	Validation accuracy
SimpleNet	62.6%	48.4%
+ Jittering	55%	29.4% (Consistently lower than row 1)
+ Zero-centering & variance-normalization	76.4%	58%
+ Dropout regularization	64.7%	57.7%
+ Making network "deep"	68.8%	57.9%
+ Batch normalization	74.8%	63.5%

Part 2: SimpleNetFinal

[Insert loss plot for SimpleNetFinal here]



[Insert accuracy plot for SimpleNetFinal here]



Final training accuracy: 74.8%

Final validation accuracy: 63.5%

Part 2: SimpleNetFinal

[Name 10 different possible transformations for data augmentation.]

1. Jittering
2. Random horizontal flip
3. Random vertical flip
4. Random rotation
5. Gaussian noise
6. Random crop
7. Random zooming
8. Random grayscale conversion
9. Random invert color
10. Random sharpness adjustment

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

Ideally, the variance after each layer should be the same. This way, every layer is learning at the same pace, which minimizes the risk of vanishing or exploding gradient.

Part 2: SimpleNetFinal

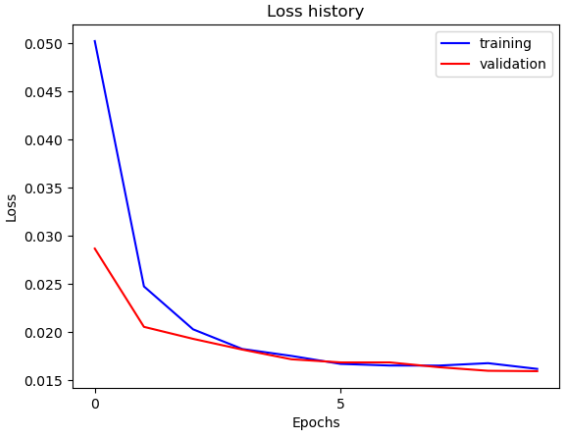
[What distribution is dropout usually sampled from?]
Bernoulli distribution.

[How many parameters does your base SimpleNet model have? How many parameters does your SimpleNetFinal model have?]
SimpleNet: 108495
SimpleNetFinal: 114650

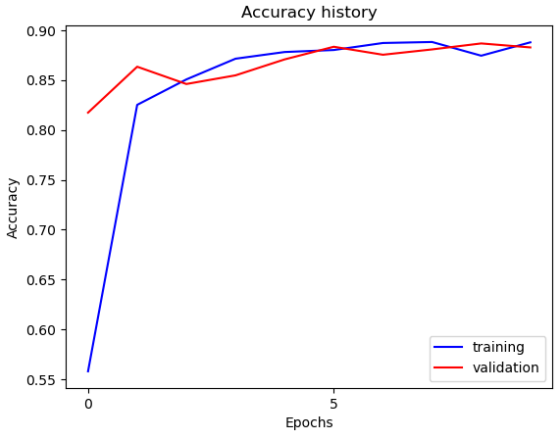
[What is the effect of batch norm after a conv layer with a bias?]
Batch normalization make training faster and more stable by re-centering and re-scaling the layer inputs. Employing batch normalization essentially removes the need to use a bias in the previous layer.

Part 3: ResNet

[Insert loss plot here]



[Insert accuracy plot here]

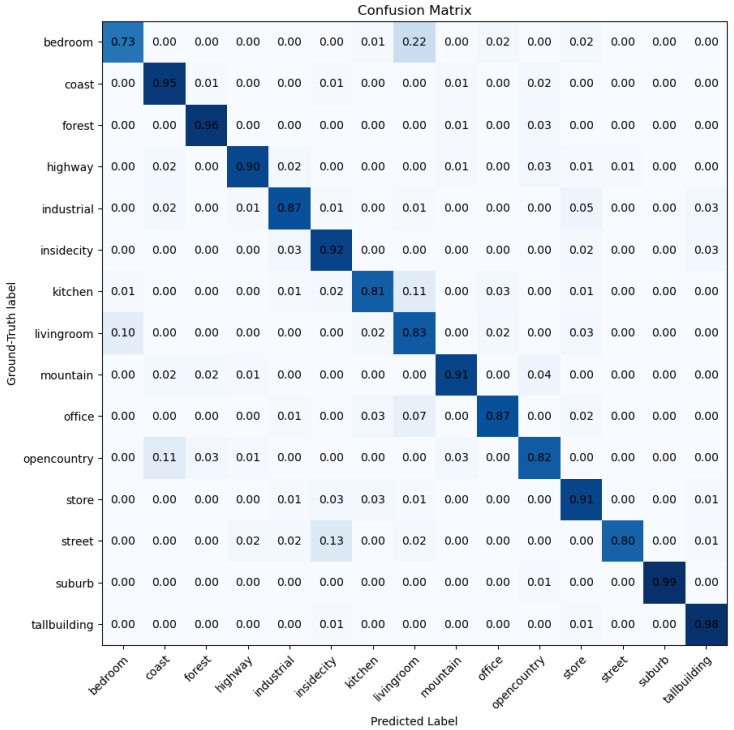


Final training accuracy: 88.7%

Final validation accuracy: 88.2%

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.]

Bedrooms misclassified as living rooms is the most misclassified class. This is likely because bedrooms and living rooms usually look similar with curtains and a soft, cushiony structure (bed or couch). In the next page, the first row shows bedrooms which were misclassified as living rooms and the second row shows properly classified living rooms.

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.]





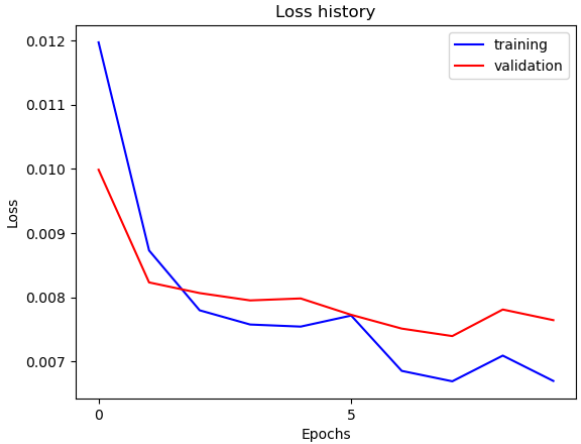

Part 3: ResNet

[What does fine-tuning a network mean?]
It means to take a pre-trained network, freeze all the parameters except for the ones in the last (few) layers and train the model on the given database to get good parameter values for the data. In other words, fine-tune an already adequate network to make it catered to the current database.

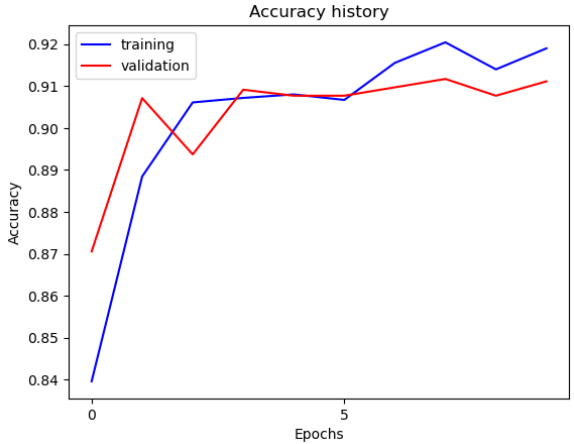
[Why do we want to "freeze" the conv layers and some of the linear layers from a pre-trained ResNet? Why can we do this?]
We want to freeze the conv layers to reduce computation as a result of training and to keep the parameter values unchanged. The conv layers learn common features from images, so the layers trained on one set of images may be generalized on another set of images.

Part 4: Multi-label Scene Attributes

[Insert loss plot here]



[Insert accuracy plot here]

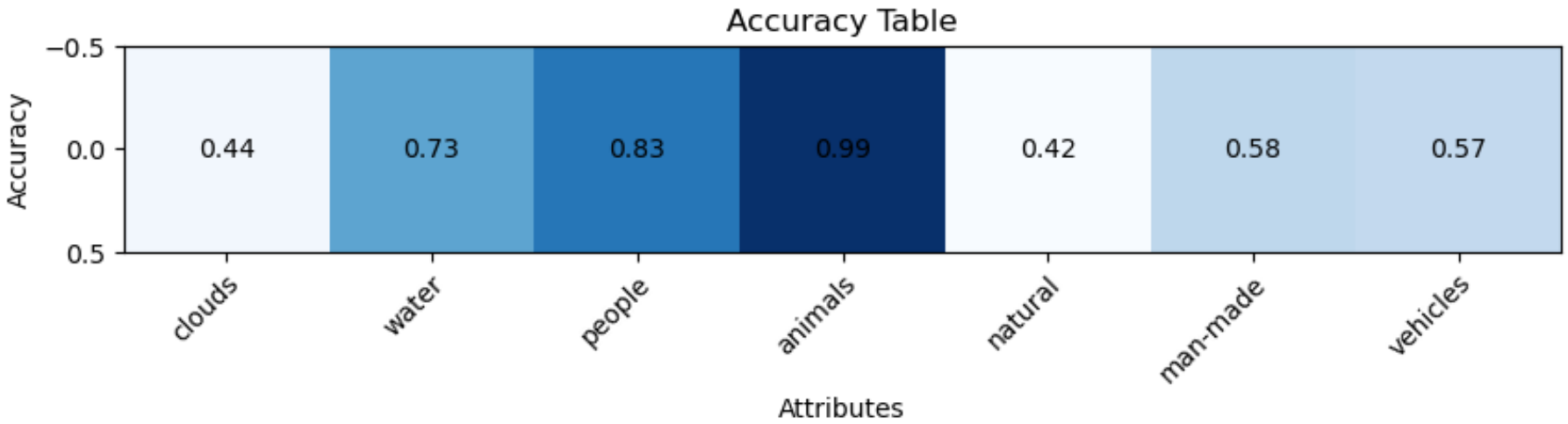


Final training accuracy: 91.9%

Final validation accuracy: 91.1%

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.]

1. Number of output features in the final linear layer changed from 15 to 7.
2. The loss function is changed from CrossEntropy to BinaryCrossEntropy.
3. A sigmoid activation function added.

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

Cross-entropy Loss is appropriate when we're classifying among more than 2 classes. Since we only wish to determine the existence/non-existence of classes in the picture, a binary cross-entropy loss is more appropriate. That being said, Cross-entropy loss still works reasonably well, yielding an accuracy of 81%.

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.]

The model may learn to associate the presence of one class due to the presence of another class. Some of the sub-classes may be separated by thin lines. For example, it's likely that a solid, rock structure beside a highway is a natural establishment, but it is just as likely to be a man-made structure beside an urban road that may look like a highway.