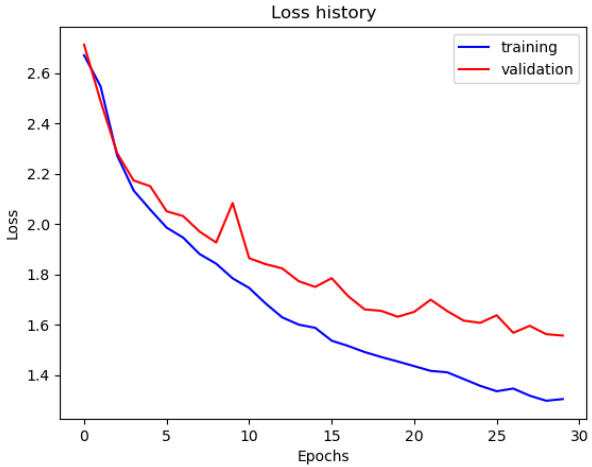
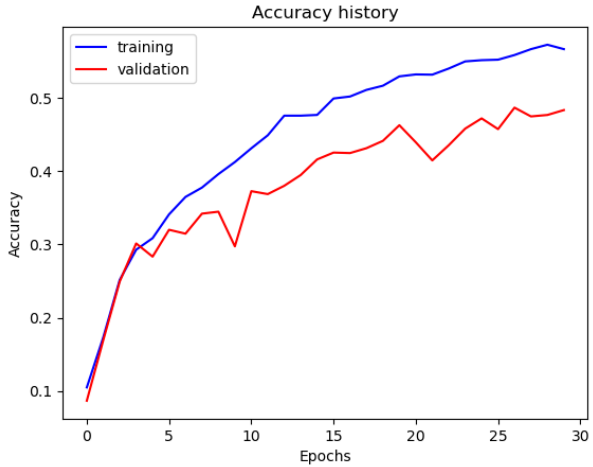
Warning: Do not delete slides.
This includes extra credit slides and any problems you do not complete.
All problems, including extra credit, must be assigned to a slide on Gradescope. Failure to follow this will result in a penalty.

CS 6476 Project 3

Auryn Yamamura
ayamamura6@gatech.edu
ayamamura6
904154249

Part 1: SimpleNet





Final training accuracy: 56.65%

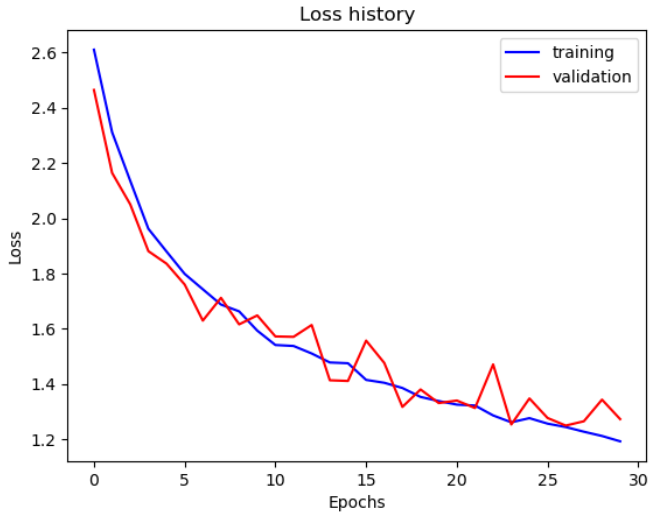
Final validation accuracy: 48.33%

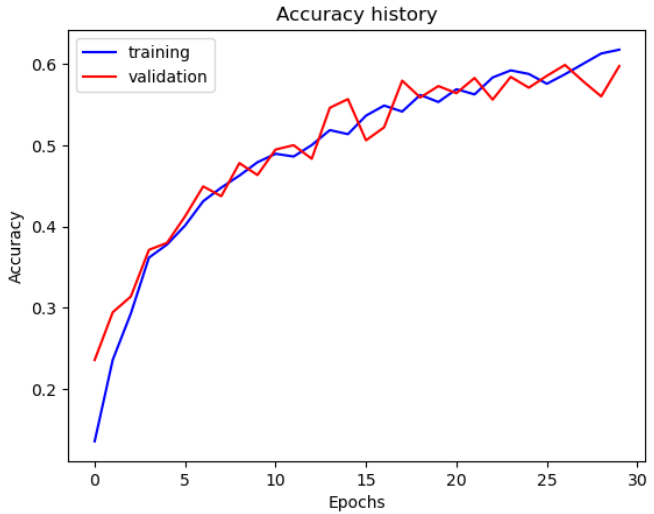
Part 2: SimpleNetFinal

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

	Training accuracy	Validation accuracy
SimpleNet	56.65%	48.33%
+ Jittering	50.65%	49.33%
+ Zero-centering & variance-normalization	61.31%	55.73%
+ Dropout regularization	58.46%	56.2%
+ Making network "deep"	54.44%	52.6%
+ Batch normalization	61.74%	59.73%

Part 2: SimpleNetFinal





Final training accuracy: 61.74%

Final validation accuracy: 59.73%

Part 2: SimpleNetFinal

[Name 10 different possible transformations for data augmentation.]

1. Color Jitter
2. Random Horizontal Flip
3. Random Vertical Flip
4. Random Perspective
5. Random Rotation
6. Random Crop
7. Random Zoom In
8. Random Invert
9. Random Adjust Sharpness
10. Random Solarize

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

In general, we would want the variance of our activations to be constant throughout the network (if we normalized our data beforehand, then this constant would be 1).

This is so our gradients neither vanish nor explode during back-propagation (vanishing gradients would mean our model never updates its weights, whereas exploding gradients mean larger updates so it is possible that 1. we take longer to converge to the minimum loss and 2. larger updates could potentially increase loss.)

Part 2: SimpleNetFinal

[What distribution is dropout usually sampled from?]

Bernoulli Distribution

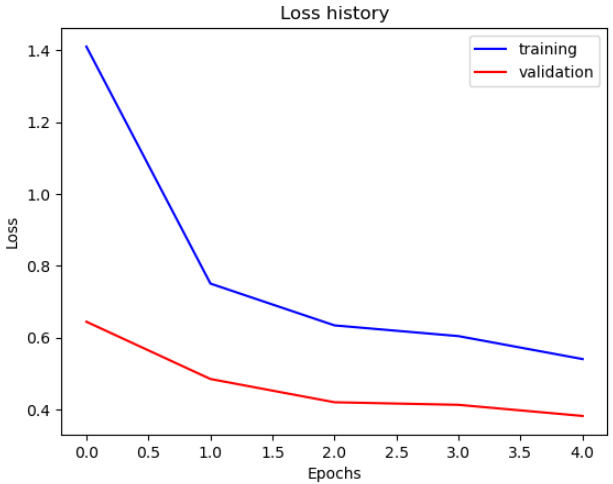
[How many parameters does your base SimpleNet model have? How many parameters does your SimpleNetFinal model have?]

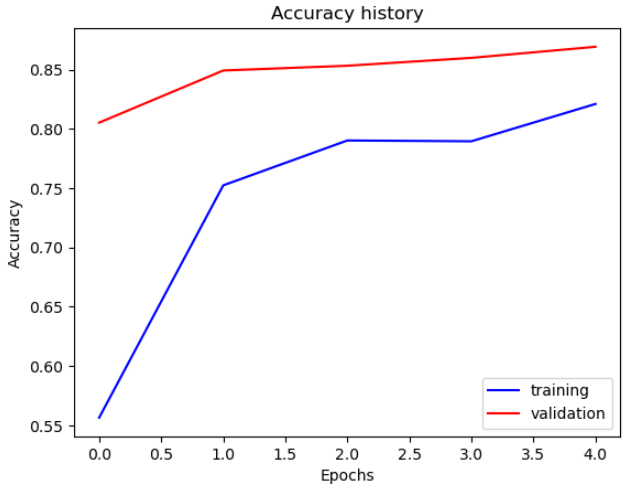
SimpleNet model has 56895 parameters.
SimpleNetFinal model has 79375 parameters.

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

The BatchNorm layer gets rid of the bias by normalizing our activations. In other words, the process of normalization shifts our activations so we get (as close as possible to) mean=0, which also implicitly sets the bias to 0 (or at least extremely to 0).

Part 3: ResNet



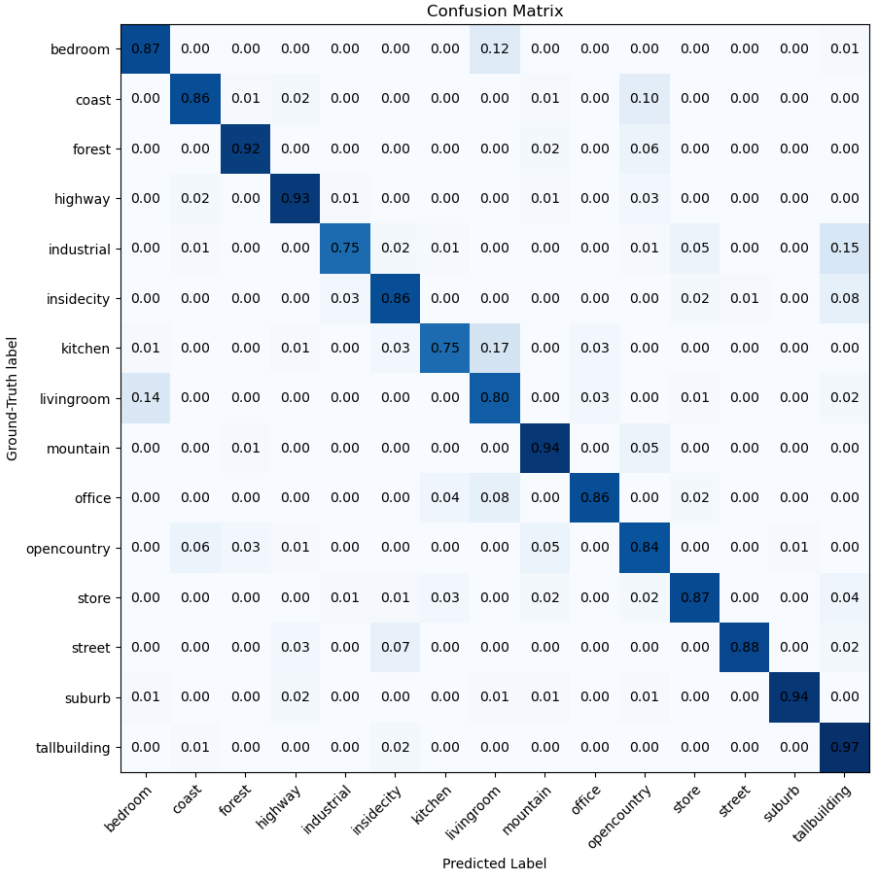


Final training accuracy: 82.11%

Final validation accuracy: 86.93%

Part 3: ResNet

[Insert visualization ...nal 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.]

The most misclassified classes for my model were ‘Industrial’ and ‘Kitchen’ with 75% accuracy each. I chose to only display misclassified images from ‘Kitchen’ since 17% of ‘Kitchen’ was misclassified as ‘LivingRoom’ compared to 15% of ‘Industrial’ being misclassified as ‘TallBuilding’.

As to why these misclassification possibly occurred, we can look at our training datasets for ‘Kitchen’ and ‘LivingRoom’. We find that many of the images in ‘LivingRoom’ train dataset contain multiple chairs (often cushioned), whereas our ‘Kitchen’ train dataset contains primarily kitchen appliances or stovetops. In all of the images above, we can see multiple chairs (with the leftmost image containing cushioned chairs). Additionally both train datasets overlap in that they have images containing dining room tables/chairs (which are present in the two rightmost images).

Part 3: ResNet

[What does fine-tuning a network mean?]

Fine-tuning is the process of adapting a pre-trained model for specific tasks or use cases. In this project, we downloaded a pre-trained network (ResNet18) and modified its fully connected layers for our specific image classification problem, freezing all other layers. Afterwards we trained the unfrozen layers on our dataset.

[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 freeze the convolution layers to preserve the pre-trained weights. To be more specific, since the task we want to accomplish and the task that the pre-trained ResNet18 was trained for are similar enough, we want to preserve pre-trained image representation/image feature detectors that ResNet18 previously learned on the ImageNet dataset.

We do not freeze the fully connected (i.e. linear) layers as the classifiers for our two tasks need to be different.

Part 4: Multi-label Scene Attributes (Extra Credit)

[Insert loss plot here]

[Insert accuracy plot here]

Final training accuracy:

Final validation accuracy:

Part 4: Multi-label Scene Attributes (Extra Credit)

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

Part 4: Multi-label Scene Attributes (Extra Credit)

[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 (Extra Credit)

[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.]

Extra credit (optional)

[Discuss what extra credit you did and provide analyses.]