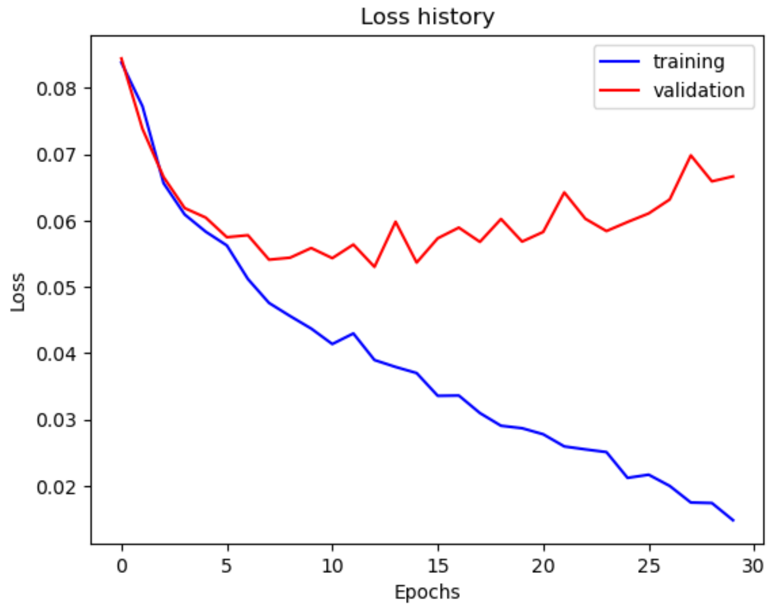
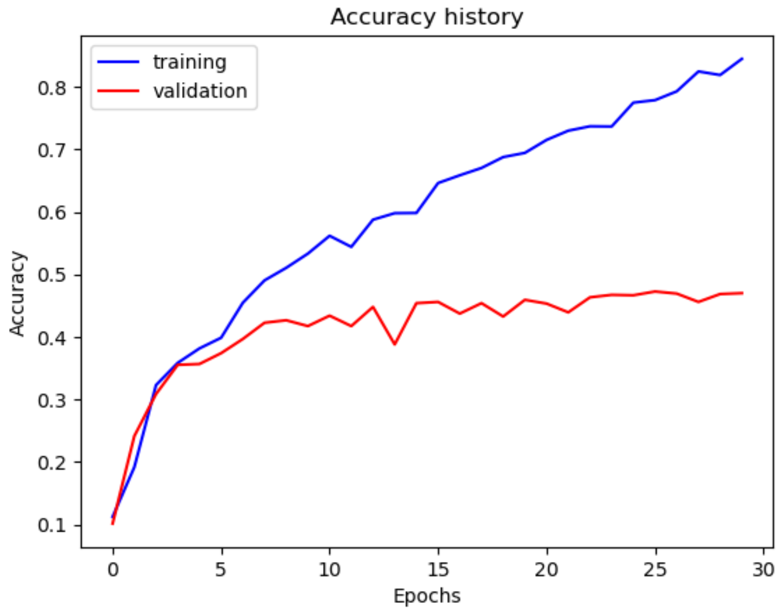
Warning: Do not delete slides.This includes extra credit slides and any problems you do not complete. All problems, including extra credit, must be assigned to a slide on Gradescope. Failure to follow this will result in a penalty.

CS 6476 Project 3

Minjun Kim
Mkim925@gatech.edu
mkim925
903873051

Part 1: SimpleNet





Final training accuracy: 0.844891122278057

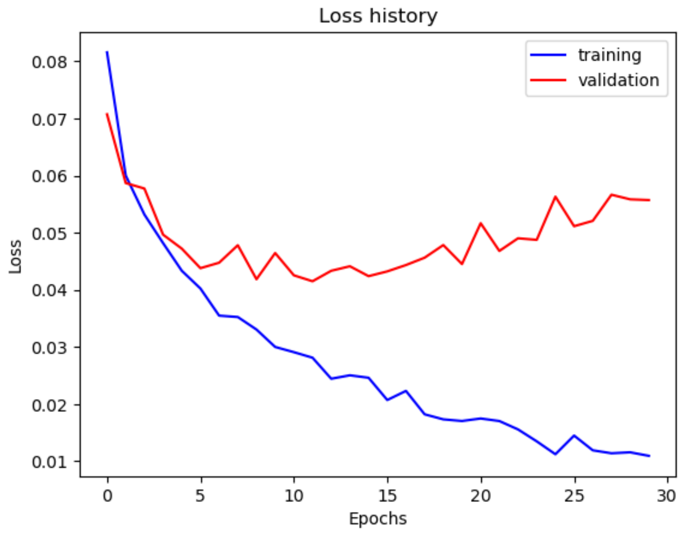
Final validation accuracy: 0.47

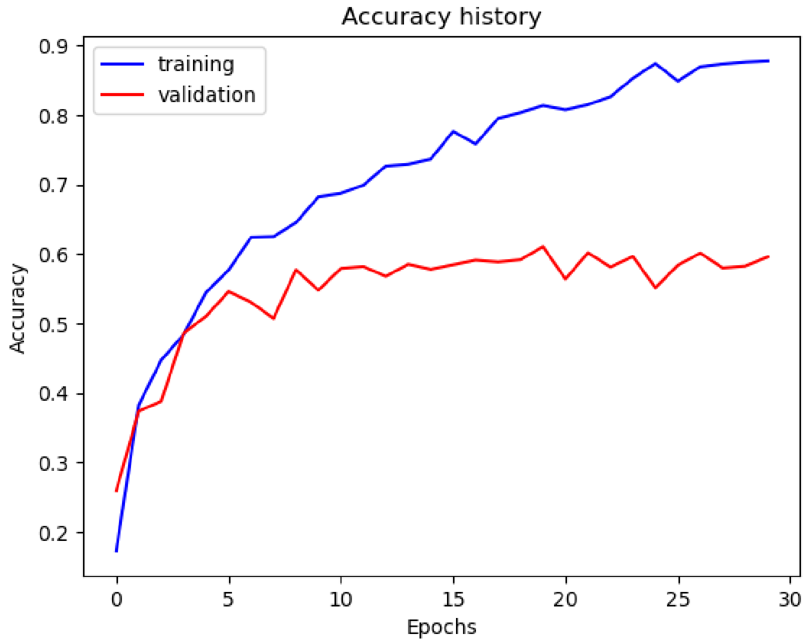
Part 2: SimpleNetFinal

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

	Training accuracy	Validation accuracy
SimpleNet	0.844891122278057	0.47
+ Jittering	0.8533	0.5367
+ Zero-centering & variance-normalization	0.9457	0.5420
+ Dropout regularization	0.9410	0.5893
+ Making network "deep"	0.8623	0.5987
+ Batch normalization	0.8781	0.5960

Part 2: SimpleNetFinal





Final training accuracy: 0.8780569514237856

Final validation accuracy: 0.596

Part 2: SimpleNetFinal

random horizontal flip, random vertical flip, random rotation, random cropping, affine transformations (which combine rotation, scale, translation, and shear), color jitter (adjusting brightness, contrast, saturation, and hue), conversion to grayscale, addition of Gaussian noise, random zoom (resized crop), and elastic or perspective distortions.

The desired variance after each layer in a neural network is typically around 1. Maintaining this unit variance is helpful because it prevents the gradients from vanishing or exploding during training, particularly in deep networks.

Part 2: SimpleNetFinal

Dropout is typically sampled from a Bernoulli distribution. For each neuron during training, a Bernoulli trial is conducted where the neuron is retained with a probability of 1−p and dropped with probability p.

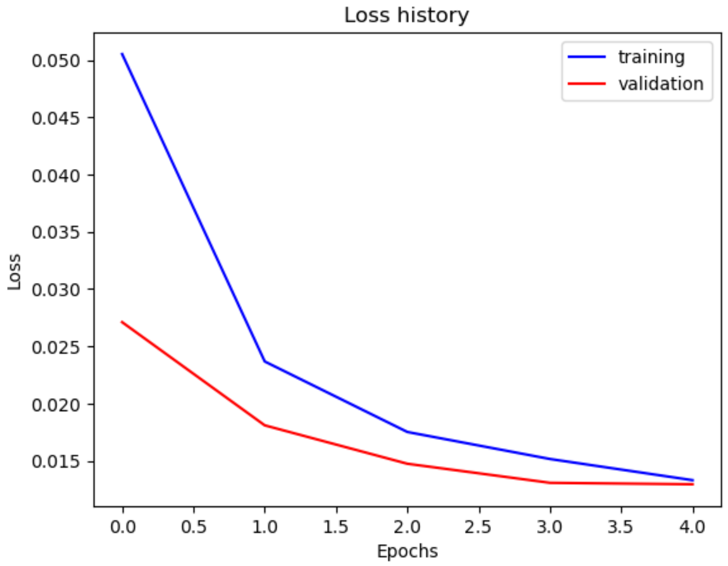
SimpleNet:
Conv1 (250 + 10) + Conv2 (5000 + 20) + FC1 (5120*128 + 128) + FC2 (128*15 + 15) = 260 + 5020 + 655488 + 1935 = **662703**

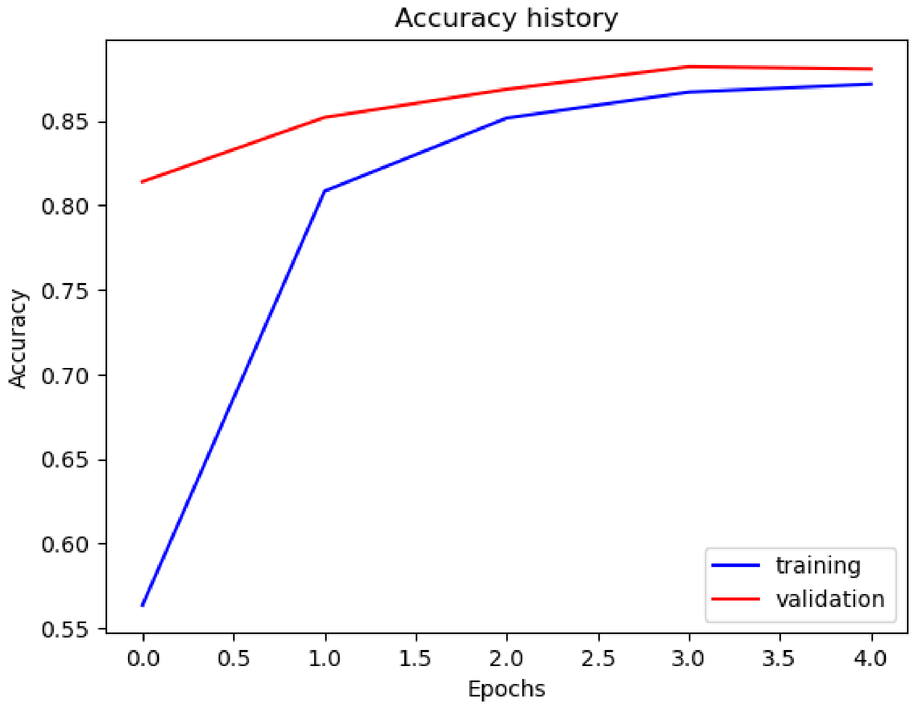
SimpleNetFinal:
Conv1 (250 + 10) + BatchNorm (10 + 10) + Conv2 (5000 + 20) + BatchNorm (20 + 20) + Conv3 (5400 + 30) + FC1 (1920*128 + 128) + FC2 (128 * 15 + 15) = 260 + 20 + 5020 + 40 + 5430 + 245888 + 1935 = 258593

*Note that biases in SimpleNetFinal are counted because we didn't disable weights, although we set them to 0 and they won't influence the output as we use batchNorm

When Batch Normalization is applied immediately after a convolutional layer that includes a bias term, the effect of that bias is effectively negated. This is because BatchNorm normalizes the output of the convolution by subtracting the batch mean and dividing by the batch standard deviation, followed by its own learnable shift (β) and scale (γ). As a result, the original bias added by the convolution becomes redundant and does not contribute meaningfully to the output.

Part 3: ResNet

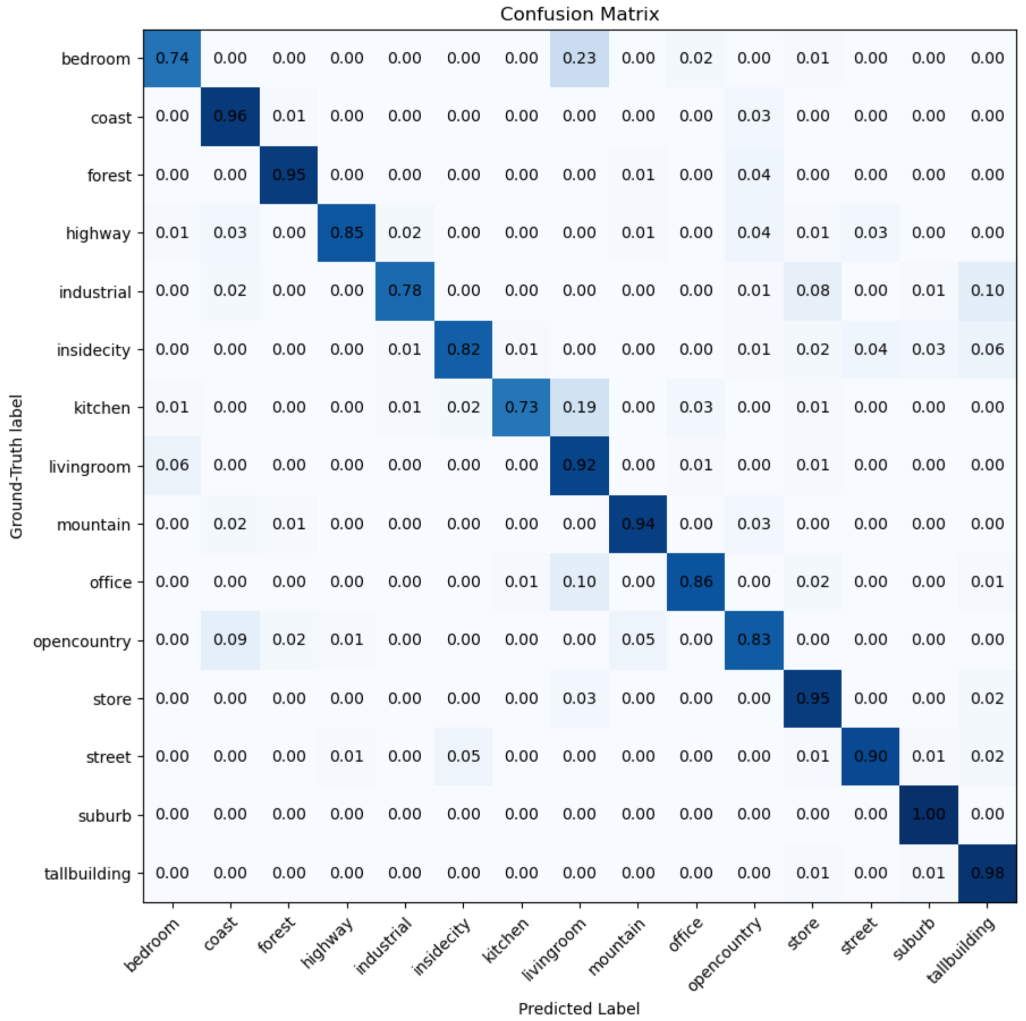




Final training accuracy: 0.8716917922948074

Final validation accuracy: 0.8806666666666667

Part 3: ResNet



	bedroom	coast	forest	highway	industrial	insidecity	kitchen	livingroom	mountain	office	opencountry	store	street	suburb	tallbuilding
bedroom	0.74	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.00	0.02	0.00	0.01	0.00	0.00	0.00
coast	0.00	0.96	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00
forest	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.04	0.00	0.00	0.00	0.00
highway	0.01	0.03	0.00	0.85	0.02	0.00	0.00	0.00	0.01	0.00	0.04	0.01	0.03	0.00	0.00
industrial	0.00	0.02	0.00	0.00	0.78	0.00	0.00	0.00	0.00	0.00	0.01	0.08	0.00	0.01	0.10
insidecity	0.00	0.00	0.00	0.00	0.01	0.82	0.01	0.00	0.00	0.00	0.01	0.02	0.04	0.03	0.06
kitchen	0.01	0.00	0.00	0.00	0.01	0.02	0.73	0.19	0.00	0.03	0.00	0.01	0.00	0.00	0.00
livingroom	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.00	0.01	0.00	0.01	0.00	0.00	0.00
mountain	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.03	0.00	0.00	0.00	0.00
office	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.10	0.00	0.86	0.00	0.02	0.00	0.00	0.01
opencountry	0.00	0.09	0.02	0.01	0.00	0.00	0.00	0.00	0.05	0.00	0.83	0.00	0.00	0.00	0.00
store	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.95	0.00	0.00	0.02
street	0.00	0.00	0.00	0.01	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.01	0.90	0.01	0.02
suburb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
tallbuilding	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.98

Part 3: ResNet

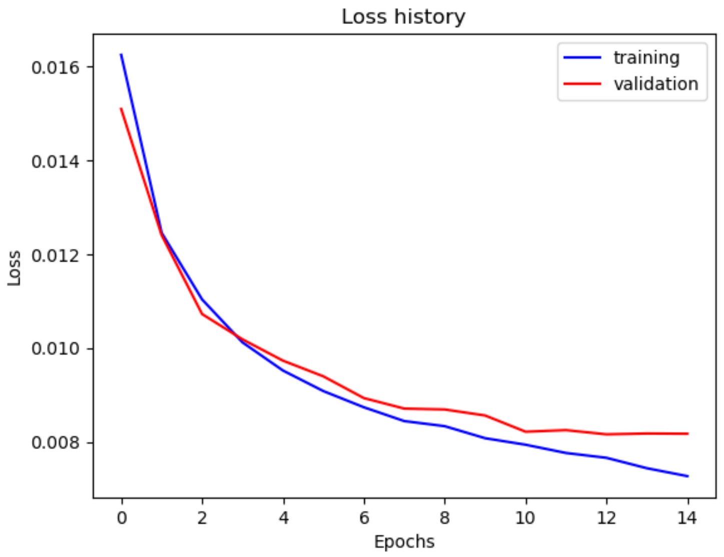
The confusion between the "livingroom" and "bedroom" classes in the confusion matrix is likely due to the significant visual and contextual overlap between the two environments. Both rooms commonly contain similar furniture such as couches, chairs, lamps, shelves, or even beds with headboards, and they often share comparable lighting conditions, color schemes, and decorative elements like carpets or curtains. This similarity becomes more pronounced when images are resized or cropped, causing the model to miss discriminative features.

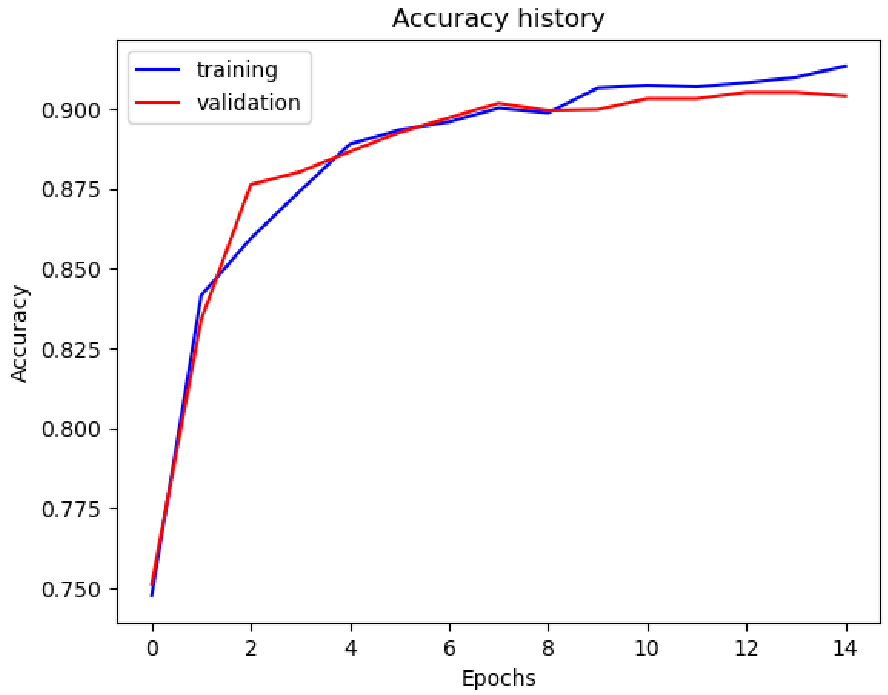
Part 3: ResNet

Fine-tuning a network refers to the process of taking a pre-trained model typically trained on a large, general-purpose dataset and adapting it to perform well on a new, often smaller and more specific task or dataset. Instead of training the entire model from scratch, which can be computationally expensive and require large amounts of data, fine-tuning allows us to leverage the learned features from earlier layers of the pre-trained model.

We want to freeze the convolutional layers and some of the linear layers from a pre-trained ResNet because these layers have already learned to extract general, low-level and mid-level features from images. By freezing them, we prevent their weights from being updated during training, which reduces the number of parameters that need to be learned and significantly speeds up training. This also helps avoid overfitting, especially when we are working with smaller datasets. We can do this because the early layers of a convolutional neural network are not specific to any particular dataset—they capture universal patterns that are useful for most image classification tasks. Only the final layers are highly task-specific, so we replace and fine-tune those to adapt the model to our new classification problem.

Part 4: Multi-label Scene Attributes

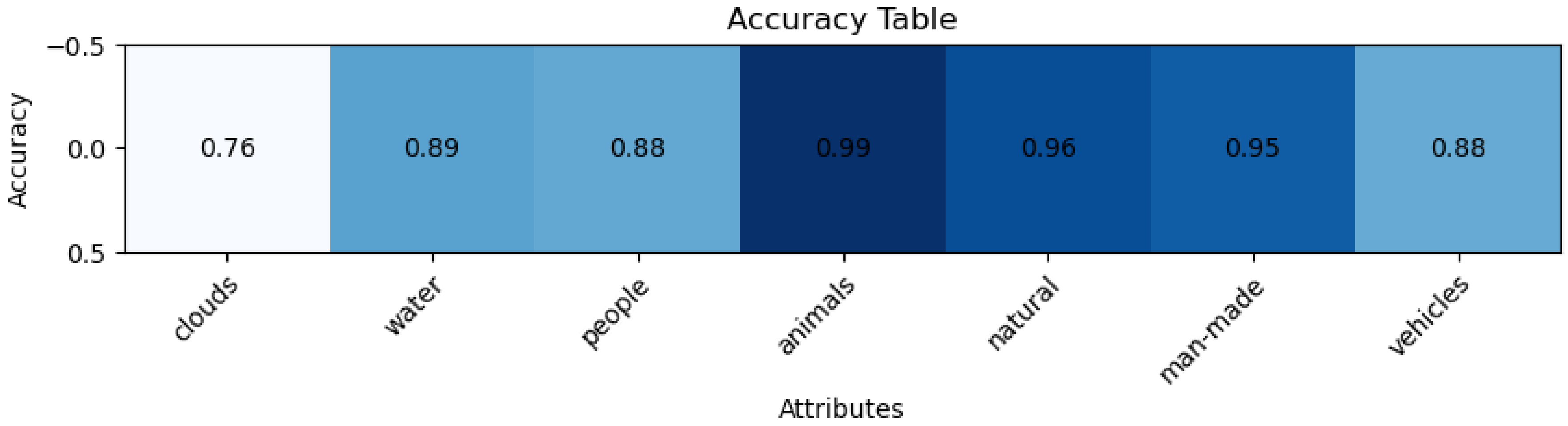




Final training accuracy: 0.9132824075660578

Final validation accuracy: 0.9040000438690186

Part 4: Multi-label Scene Attributes



Part 4: Multi-label Scene Attributes

1. Changed loss function from CrossEntropyLoss into BCELoss since we have binary labels now.
2. Added Sigmoid activation function
3. Our output shape now has 7 attributes instead of 15.

*Note that I used BCEWithLogitsLoss instead of BCELoss & Sigmoid activation function independently as BCEWithLogitsLoss is safer in terms of numerical stability.

No. CrossEntropyLoss is suitable for single-label classification, where each image belongs to exactly one class out of many mutually exlclusive classes. However, in part 4 we instead use BCELoss (Binary Cross Entropy) because each image can belong to multiple classes. We now have seven independent binary classification problems instead of one multi-class problem.

Part 4: Multi-label Scene Attributes

A key problem to be wary of in multi-label classification is class imbalance, where some attributes (labels) appear far more frequently than others in the dataset. This imbalance can lead to a model that performs well overall (e.g., >90% average accuracy) but fails to correctly learn and predict the minority labels.

There is a possibility of high overall accuracy but poor per-label accuracy. This is why we visualize the results with an accuracy table: it shows performance per attribute, allowing us to detect if certain classes are being under-predicted or ignored entirely.

In our accuracy table, we can see that the 'clouds' attribute has a much lower accuracy of 0.76 compared to the overall accuracy of ~90%. This suggests that the model is not equally effective across all labels, likely because some attributes appear more frequently in the dataset, making them easier for the model to learn. Less frequent labels like "clouds" may be underrepresented, leading to lower accuracy.

Extra credit (optional)

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