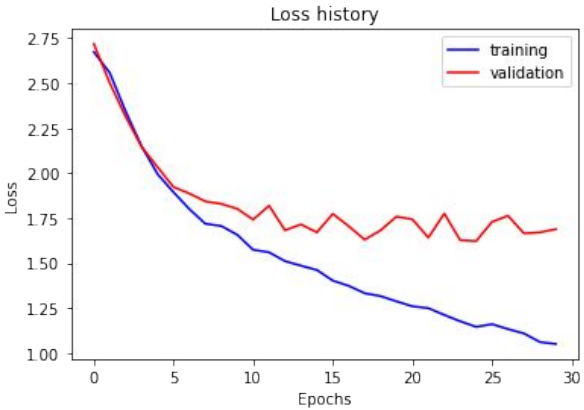
DL for Recognition

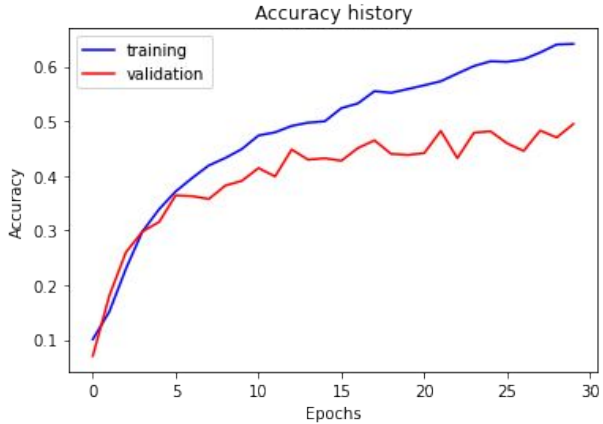
Gautham Gururajan

Part 1: SimpleNet

Loss plot for SimpleNet



Åccuracy plot for SimpleNet



Final training accuracy: 64.12060301507537%

Final validation accuracy: 49.4666666666666%

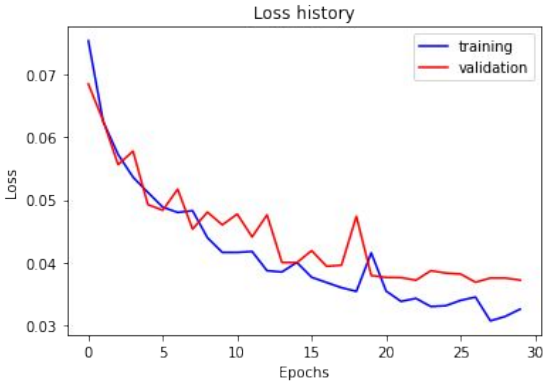
Part 2: SimpleNetFinal

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

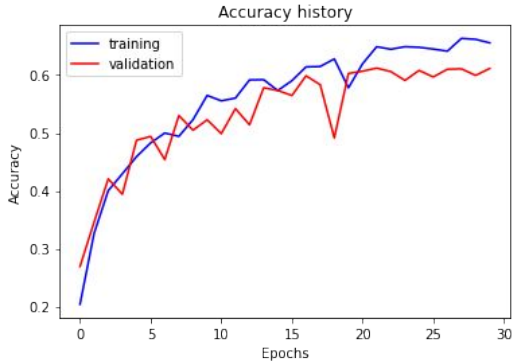
	Training accuracy	Validation accuracy
SimpleNet	64.12%	49.46%
+ Jittering	63.21%	51.32%
+ Zero-centering & variance-normalization	65.14%	56.12%
+ Dropout regularization	65.00%	58.99%
+ Making network "deep"	67.49%	60.19%
+ Batch normalization	67.56%	61.13%

Part 2: SimpleNetFinal

Loss plot for SimpleNetFinal



Accuracy plot for SimpleNetFinal



Final training accuracy: 67.56113902847571%

Final validation accuracy: 61.1333333333333%

Part 2: SimpleNetFinal

10 different possible transformations for data augmentation

1. CenterCrop
2. Normalize
3. ColorJitter
4. FiveCrop
5. Grayscale
6. Pad
7. RandomHorizontalFlip
8. RandomResizedCrop
9. RandomRotation
10. RandomVerticalFlip

Desired variance after each layer?

We would like to have unit variance after each layer. This would be useful as it constrains the gradients of each layer to be on a similar scale. This in turn allows your model to learn faster (The objective of batch norm is exactly this).

Part 2: SimpleNetFinal

What distribution is dropout usually sampled from?

- Dropout is sampled from the Bernouli distribution. It randomly drops input nodes with probability ‘p’ to help avoid overfitting.

How many parameters does the base SimpleNet model have? How many parameters does your SimpleNetFinal model have?

- SimpleNet : 56895
- SimpleNetFinal : 45210

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

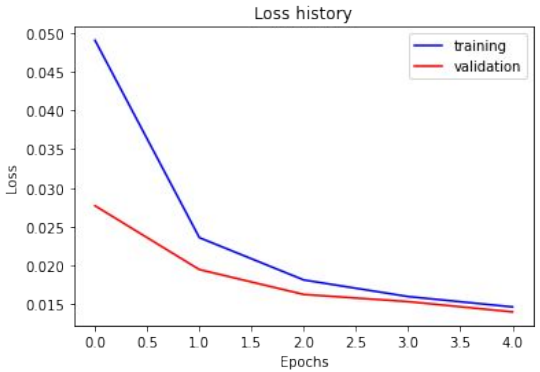
Batch norm is the same irrespective of whether or not a conv layer has a bias term as it already takes into account a “bias”.

In general, Batch norm allows us to truncate values in our conv layer (Normalization). This helps in the following ways -

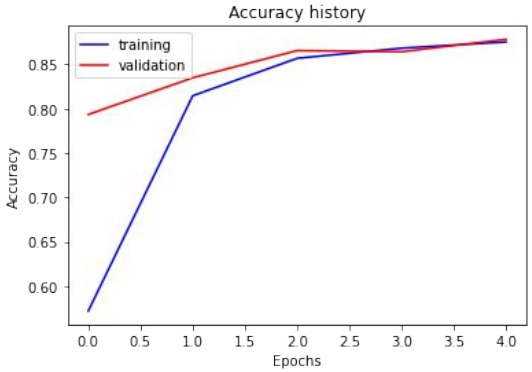
1. It “truncates” the distribution which means that the standard deviation (and hence variance) is reduced and hence removes the problem of poor conditioning.
2. This means that it regularizes the layer, which implies that the gradients do not vary as greatly as before.
3. This directly improves speed of training, decreases dependance on initial weights and helps to saturate forms of non-linearity.

Part 3: ResNet

Loss plot



Accuracy plot

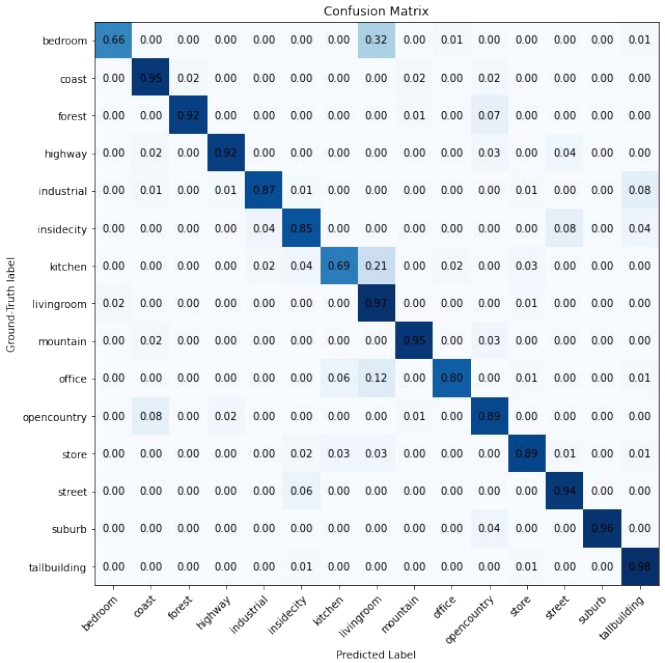


Final training accuracy: 87.50418760469012%

Final validation accuracy: 87.8%

Part 3: ResNet

Visualization of confusion matrix obtained from the final ResNet model



Part 3: ResNet

Visualizations of 3 misclassified images from the most misclassified class according to your confusion matrix.

- Using our confusion matrix, we see that the bedroom is most misclassified (and living room is the highest misclassified class for bedroom). Using ‘get_pred_images_for_target’ we retrieve misclassified images. 3 of them are as seen below. The reason as to why they are misclassified is that they share many attributes with “Living room”. As you can see, all the images have sofas or televisions. These attributes correspond strongly to living rooms rather than bedrooms and hence the misclassification.



Part 3: ResNet

What does fine-tuning a network mean?

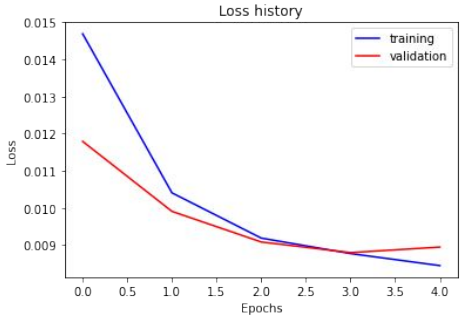
- Fine-tuning is a process that uses a model that has already been trained once before (for a given task, similar to the current task) and then ‘tunes’ the model to make it perform for the current task.

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?

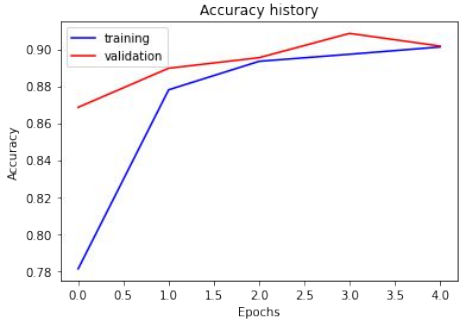
- Freezing the layer means that corresponding weights are not trained.
- This helps in computation time, as a huge order of weights do not need to be updated.
- This works because of the context that the tasks are similar. For similar tasks, we need not change the weights of the layers towards the middle since there is a good chance that they capture very similar information. Instead, we change the last layers to “fine tune” our model and steer it towards the current task. The more unsimilar the tasks are, the more we will need to ‘unfreeze’

Part 4: Multi-label Scene Attributes

Loss plot



Accuracy plot

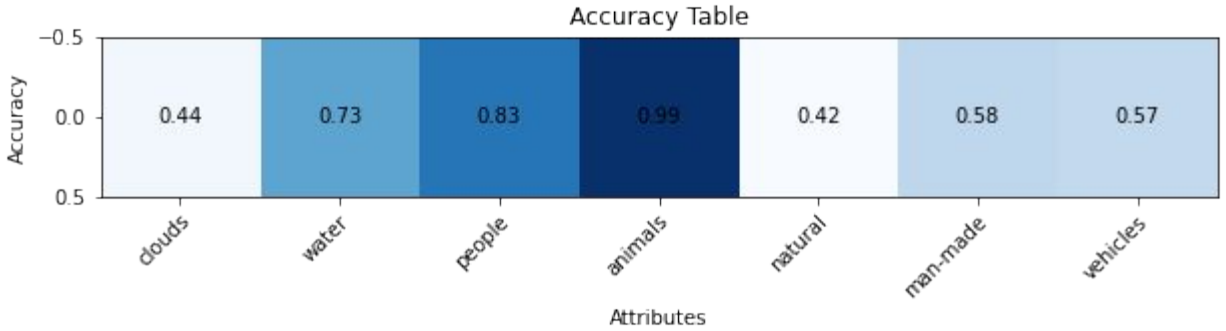


Final training accuracy: 90.12183468705208%

Final validation accuracy: 90.1714285714285%

Part 4: Multi-label Scene Attributes

Visualization of accuracy table obtained from the final MultilabelResNet model.



Part 4: Multi-label Scene Attributes

3 changes that I made in the network compared to the one in part 3

Three changes made are :

- Output is changed from size of 15 to size of 7.
- Loss function is changed from Cross Entropy to Binary Cross Entropy.
- Sigmoid activation is applied to the final output in the forward call.

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

For part 3 (Multi-class classification), we use cross entropy (CE) loss and for part 4 (Multi-label classification), we use binary cross entropy (BCE) loss.

For multi-class, we use CE - which takes in only non 0/1 inputs as it is Sum(-p*log(p)) (0 and 1 will give loss as 0, undefined and will hence cause errors during any summation).

For multi-label, we want all our output to be in the form of a probability (We apply sigmoid to this and use a threshold and accordingly assign a class by giving only 0/1 outputs)). Hence, BCE is more relevant when using a sigmoid and threshold.

Part 4: Multi-label Scene Attributes

A problem that one needs to be wary of with multilabel classification

One big problem that we see through EDA is that there can be class imbalance. This means that there are more examples from one class from the others in the training set. For example, in a binary classification, if we have only 1 example from class 1 and 99 examples from class 2, we can have our model to just predict class 2 for every input. This will give us an accuracy of 99% which seems high, but the model is clearly incorrect. This is also one reason why using accuracy table is not the best way of visualizing success. A better way to do so would be seeing precision/recall from the confusion matrix instead.