CNN Part A – Noise Layer and Batch Normalization

1.Noise Layers

Adding noise is another way to prevent a neural network from 'learning' the training data. Noise is a matrix containing small random values (different on each training iteration), which are added to the outputs of a layer. This way consequent layers will not be able to co-adapt too much to the outputs of the previous layers.

Keras has three implementations of the noise layers:

- **GaussianNoise**: general noise layer, which adds zero-centered noise, with specified standard deviation;
- **GaussianDropout**: a combination of Dropout and 1-centered Gaussian noise; the rate specifies the percentage of units to be dropped, and the standard deviation of the noise is calculated as sqrt(dropout_rate / (1 — dropout_rate)).
-
- **AlphaDropout :** Alpha Dropout is a Dropout that keeps mean and variance of inputs to their original values, in order to ensure the self-normalizing property even after this dropout. Alpha Dropout fits well to Scaled Exponential Linear Units by randomly setting activations to the negative saturation value.

Dropout

If you already have some experience with neural networks, you may already know that the main idea of dropout is to randomly turn-off some of the units in a neural network on each iteration of the training together with all its input and output connections, so that they are not affected by the gradient updates:

What to keep in mind when using Dropout

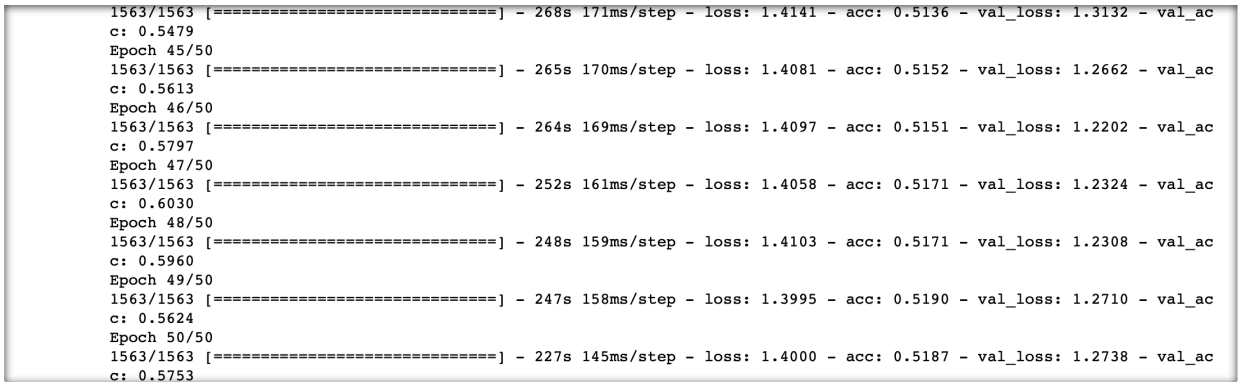
- **Network Size.** Dropping units reduces the capacity of a neural network. If n is the number of hidden units in any layer and p is the dropout rate, then after dropout only pn units will remain. Therefore, if an n-sized layer is optimal for a

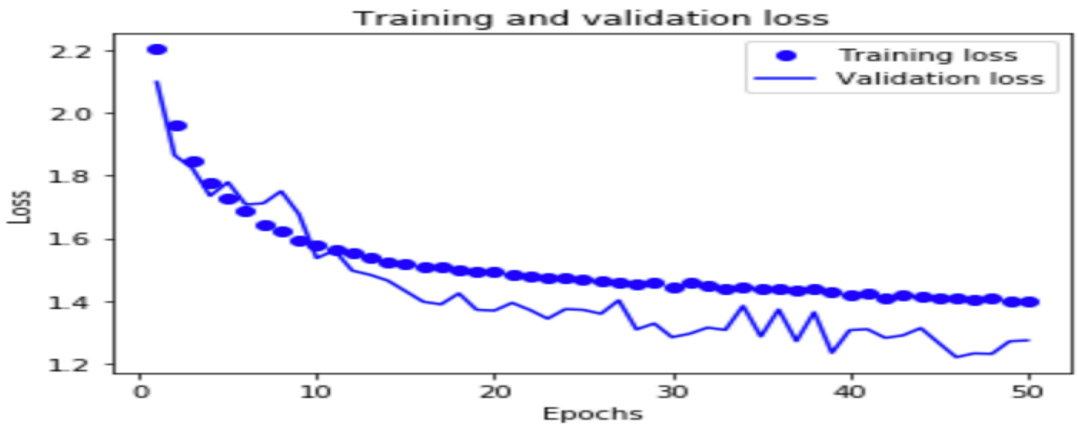
standard neural net on a given task, a good dropout net should have at least n/(1 — p) units.

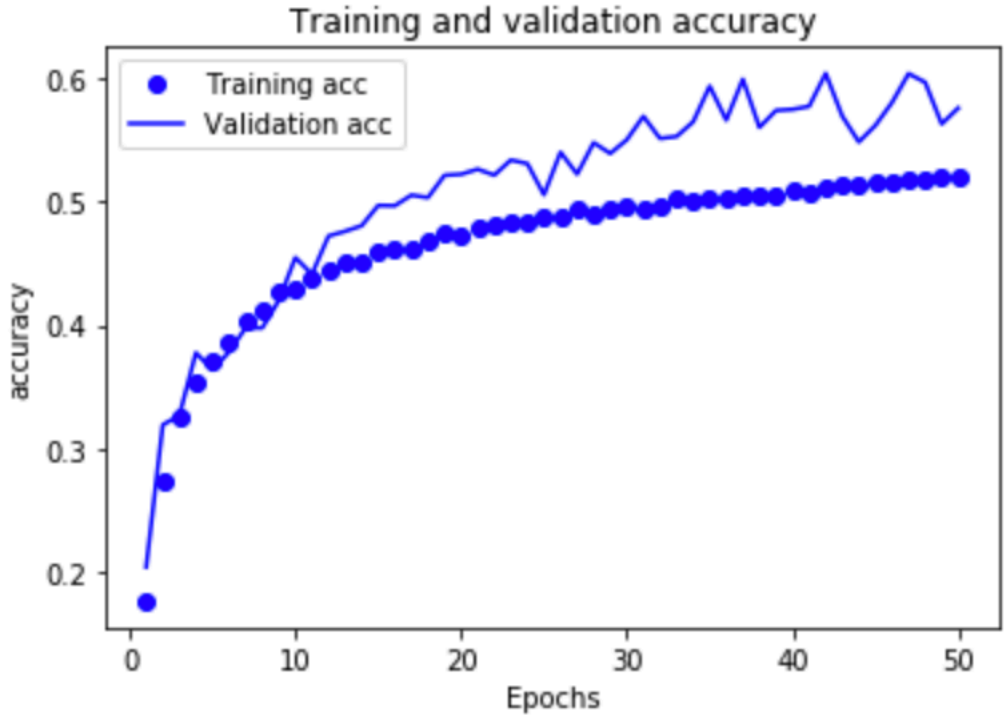
- **Learning Rate and Momentum.** Dropout introduces a significant amount of noise in the gradients compared to standard stochastic gradient descent. Therefore, a lot of gradients tend to cancel each other. In order to make up for this, a dropout net should typically use 10–100 times the learning rate that was optimal for a standard neural net or to use a high momentum. While momentum values of 0.9 are common for standard nets, with dropout values around 0.95 to 0.99 work quite a lot better. Using high learning rate and/or momentum significantly speed up learning.

Experimenting with Noise Layer – Model 2 in Notebook

As the base model was underfitting, noise layers is used as regularization layer which prevents the model to overfit. It was bound to decrease the accuracy




Training and validation loss
Training loss
Validation loss
Loss
Epochs
2.2
2.0
1.8
1.6
1.4
1.2
0
10
20
30
40
50


Training and validation accuracy
Training acc
Validation acc
accuracy
Epochs
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50

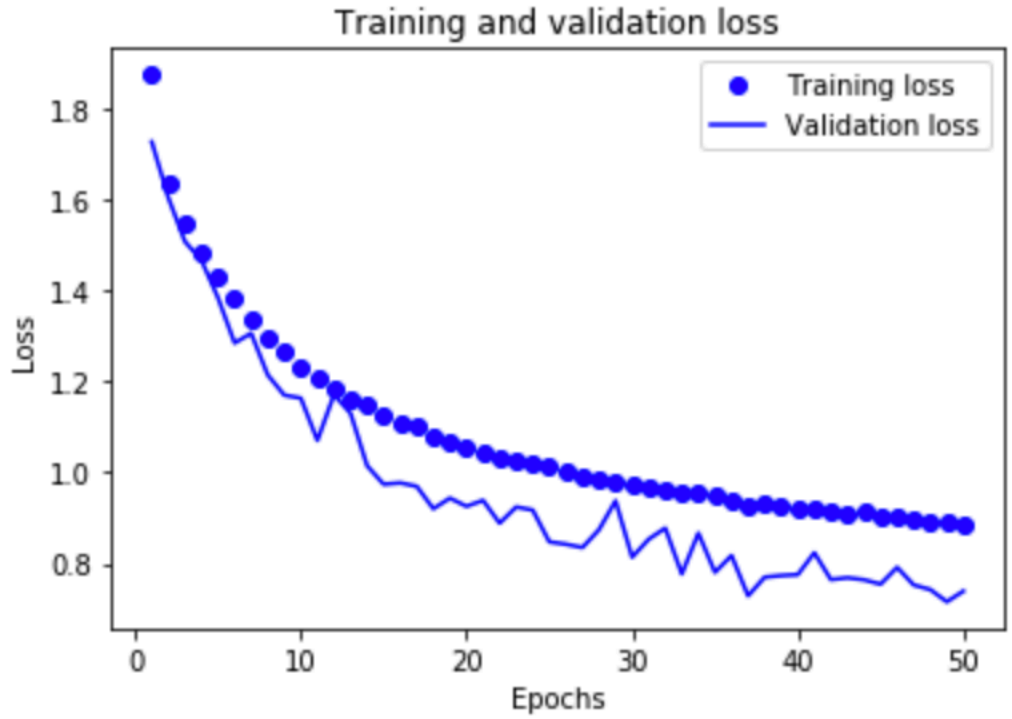
2. Batch Normalization

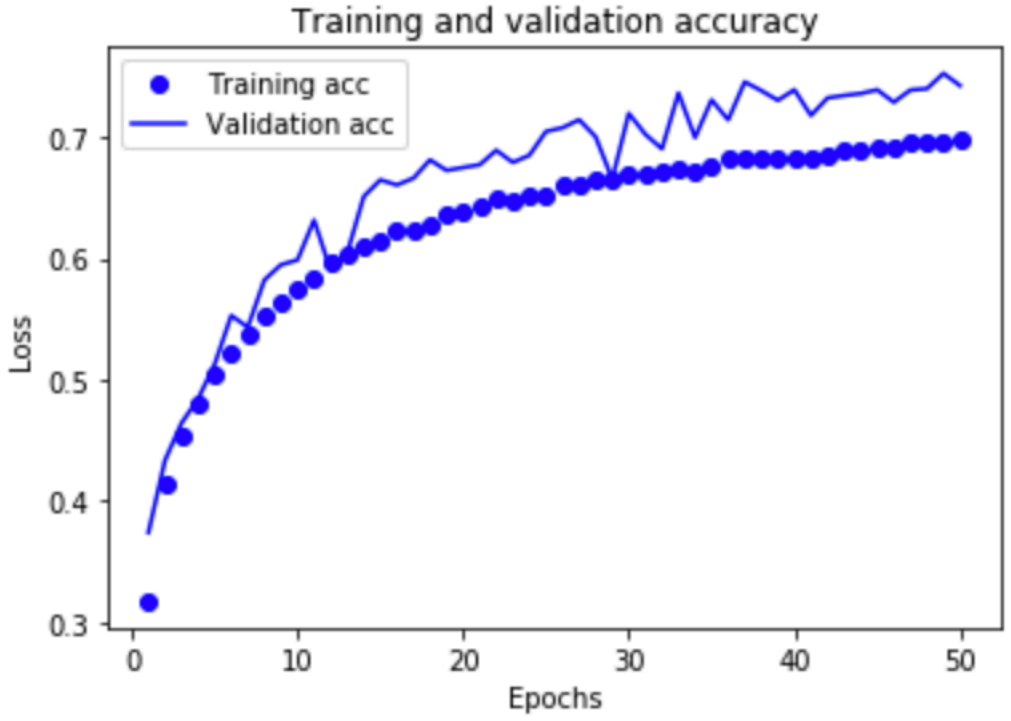
Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization,vmakes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as internal covariate shift. This address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch. Batch Normalization allows us to use much higher learning rates and be less careful about initialization.

Experimenting with Batch Normalization – Model 3 in Notebook

In this Model, adding a batch normalization layer - activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

```
c: 0.7390
Epoch 46/50
1563/1563 [==============================] - 391s 250ms/step - loss: 0.9000 - acc: 0.6905 - val_loss: 0.7913 - val_ac
c: 0.7287
Epoch 47/50
1563/1563 [==============================] - 388s 248ms/step - loss: 0.8953 - acc: 0.6953 - val_loss: 0.7525 - val_ac
c: 0.7388
Epoch 48/50
1563/1563 [==============================] - 396s 253ms/step - loss: 0.8906 - acc: 0.6949 - val_loss: 0.7424 - val_ac
c: 0.7399
Epoch 49/50
1563/1563 [==============================] - 387s 248ms/step - loss: 0.8924 - acc: 0.6947 - val_loss: 0.7156 - val_ac
c: 0.7525
Epoch 50/50
1563/1563 [==============================] - 396s 254ms/step - loss: 0.8855 - acc: 0.6986 - val_loss: 0.7391 - val_ac
c: 0.7424
Saved trained model at /Users/haroonperveez/AI/Assignment 1 Ciraf/Assignment Final /saved_models/keras_cifar10_traine
d_model3.h5
10000/10000 [==============================] - 27s 3ms/step
Test loss: 0.7390665467262268
```


Training and validation loss
Training loss
Validation loss
Loss
Epochs
1.8
1.6
1.4
1.2
1.0
0.8
0
10
20
30
40
50


Training and validation accuracy
Training acc
Validation acc
Loss
Epochs
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50

Compared to our base model, there wasn't any significant change by using batch normalization for this model.

Reference:

Noise Layers

1. https://keras.io/layers/noise/
2. https://medium.com/@maksutov.rn/deep-study-of-a-not-very-deep-neural-network-part-5-dropout-and-noise-29d980ece933

Batch Normalization

3. https://keras.io/layers/normalization/
4. https://arxiv.org/abs/1502.03167