Assignment 2

Data Preprocessing

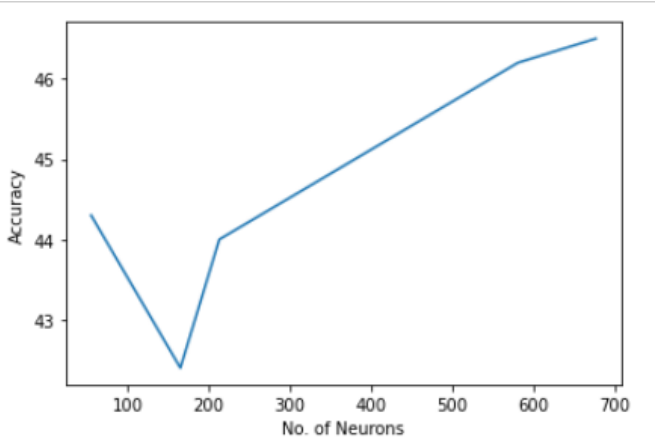
- The images are resized into dimensions of 32x32x3, to make the computations much faster. Instead of using 128x128x3 (nearly 50 thousand), we new resized our input into 32x32x3 (nearly 3 thousand).
- I flattened the images to be fed to the neural network
- I normalized the images by subtracting the mean and dividing by the standard deviation

Hyperparameter tuning

Model architecture

These are the accuracy and loss of the validation data when tested on these architectures.

Architecture	No.of Neurons	acc	loss
[50, 5]	55	0.443	1.33
[128, 32, 5]	165	0.424	1.313
[512, 64, 5]	581	0.462	1.25
[128, 64, 16, 5]	213	0.440	1.27
[512, 128, 32, 5]	677	0.465	1.26



We can see that as the number of neurons increases, the accuracy increases. Although the first architecture has small number of neurons it gives good results with respect to the other architectures. Here, we will use the 3 layered architecture [512, 64, 5], as it has pretty good accuracy and it's only 3 layers. We will use it next

Learning rate and Regularization

I did a course search at the beginning with regularization values (10e-5 to 10e3) and learning rate values (10-3 to 10-6). They are randomly generated for 100 iterations.

Results showed higher accuracy and lower loss for learning rate values in range (10e-3, 10e-5) and regularization of (10e-1, 10e-5).

Best learning rate was 0.00075 and best regularization was 3.9e-5

Activation Functions

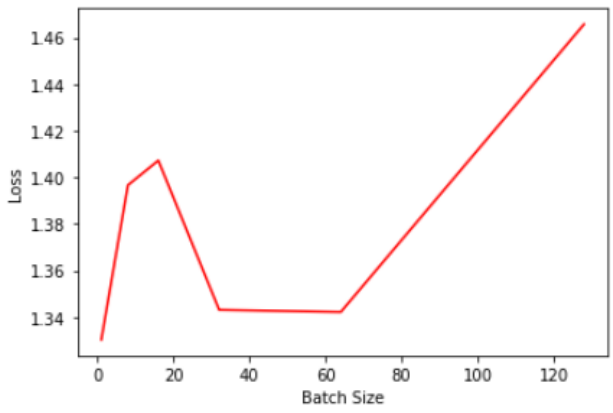
```
Activation: linear  -> loss: 1.7387127447481594 , acc: 0.34591194968553457
----------------------------------------------------------------
Activation: tanh  -> loss: 1.3688348179517722 , acc: 0.389937106918239
----------------------------------------------------------------
Activation: sigmoid  -> loss: 1.3607459590096083 , acc: 0.3490566037735849
----------------------------------------------------------------
Activation: relu  -> loss: 1.3980831052285723 , acc: 0.4119496855345912
----------------------------------------------------------------
```

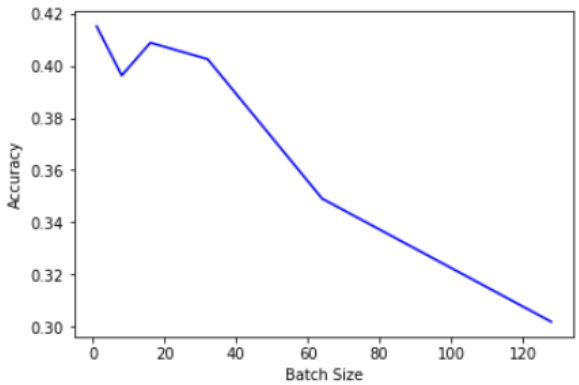
We can see that the best accuracy and loss are for relu, which we will use next

Batch Size

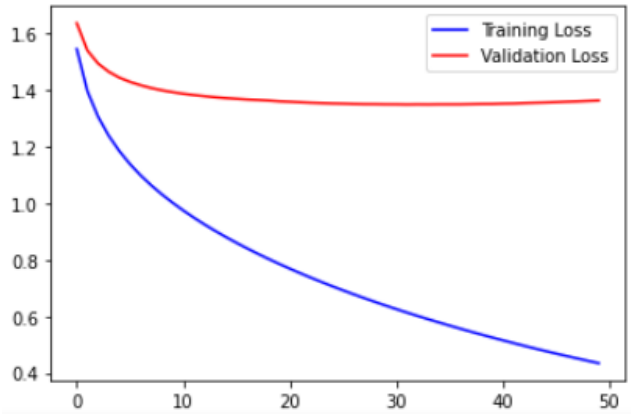
```
Batch Size: 1  -> loss: 1.330325196433499 , acc: 0.41509433962264153
----------------------------------------------------------------
Batch Size: 8  -> loss: 1.3966261554079449 , acc: 0.39622641509433965
----------------------------------------------------------------
Batch Size: 16  -> loss: 1.4072254378297335 , acc: 0.4088050314465409
----------------------------------------------------------------
Batch Size: 32  -> loss: 1.3431116247258845 , acc: 0.4025157232704403
----------------------------------------------------------------
Batch Size: 64  -> loss: 1.3421707288265932 , acc: 0.3490566037735849
----------------------------------------------------------------
Batch Size: 128  -> loss: 1.4656962990435187 , acc: 0.3018867924528302
----------------------------------------------------------------
```

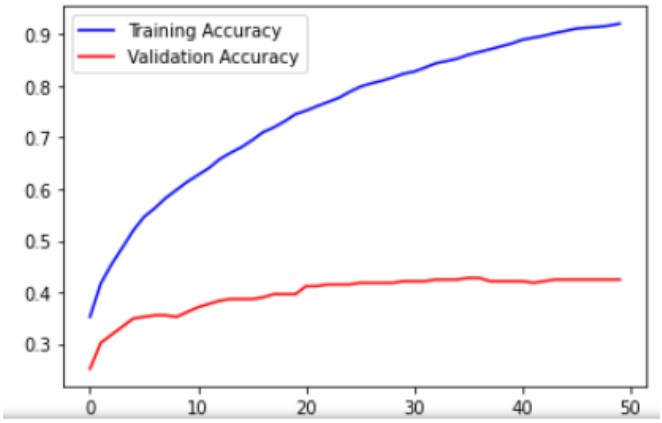
We can see that as the size of the batch increases the accuracy decreases. However, the problem with low batch_sizes is that they take a lot of time as they are less parallelizable. In our case, we will choose Batch Size of 32, as it has good accuracy besides the fact it's quite large.





Training the Neural Network





Starting epoch 20, there was little improvement in the accuracy and loss. It stops improving at this point. I stopped the training at epoch 35, because no improvement is there and even the validation accuracy begins slightly to decrease and the validation loss begins to increase. The Training loss at this point was 0.5675 and accuracy was 86% while the validation loss is 1.35 and accuracy was 42.77%.

Comparison

CCR per class

	KNN	LLS	NN
0	0.05	0.31	0.38
1	0.82	0.61	0.66
2	0.2424	0.2626	0.2626
3	0.1717	0.2424	0.6767
4	0.0909	0.2525	0.4747

AVG CCR

	KNN	LLS	NN
ACCR	0.276	0.336	0.49