Deep Learning Lab Report

Assignment 2

Ravin Kohli

Objective- Train CNNs on MNIST dataset and find best configuration via random search

We were tasked with training a Convolutional Neural Network on MNIST dataset. We had to define layers of the network and train it. Then we were supposed to run different combinations of learning rate and filters manually and plot them. As part of the exercise, we were also tasked with performing a random search to find the best possible configuration and train the network with it.

Network Architecture

The network consists of 2 Convolutional Layers, 2 Max Pooling layers, 2 Fully Connected Layers and a Softmax output.

Layer number	Number of Units/Filters	Activation Function	Size
Conv1	16	ReLu	3
Max Pooling	NA	NA	2
Conv1	16	ReLu	3
Max Pooling	NA	NA	2
Fully Connected 1	1024	ReLu	NA
Fully Connected 2	10	Sigmoid	NA

Result with Vanilla Network

Parameter	Value
Learning Rate	0.001
Number of Filters	16
Batch Size	128
Epochs	12

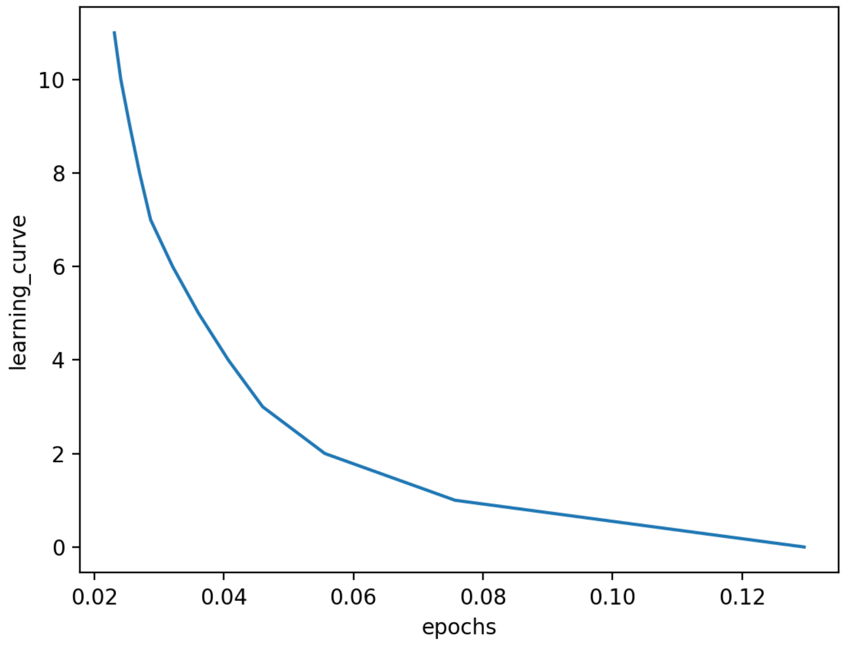

Fig 1. Loss vs Epochs

Result - The network achieved a final test error of 2.28%.

Experiments with learning rate

Results for various Learning rates are in the table below

Learning Rate	Test Error
0.1	88.65%
0.01	1.12%
0.001	2.32%
0.0001	8.98%

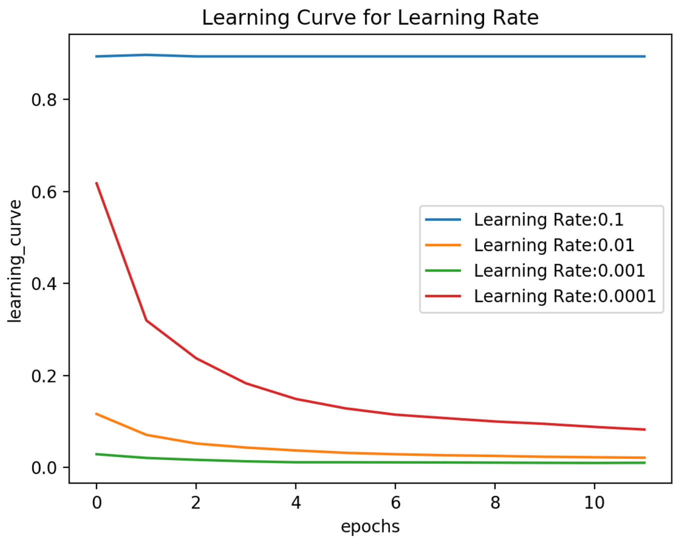

Fig 2. Loss vs Epochs

From the figure 2, we can make a number of observations.

1. With high learning rate, the model does not converge at all. (lr = 0.1, the error remains 88%).
2. Decreasing the learning rate does help the network to converge, however, we need to be careful as too low a learning rate can make the network really slow. (lr = 0.0001, the error is still 8 times the error with lr =0.01)

Experiments with Filter Size

Results for the various Filter Sizes are in the table below

Filter Size	Test Error
1	10.84%
3	2.56%
5	2.05%
7	1.65%

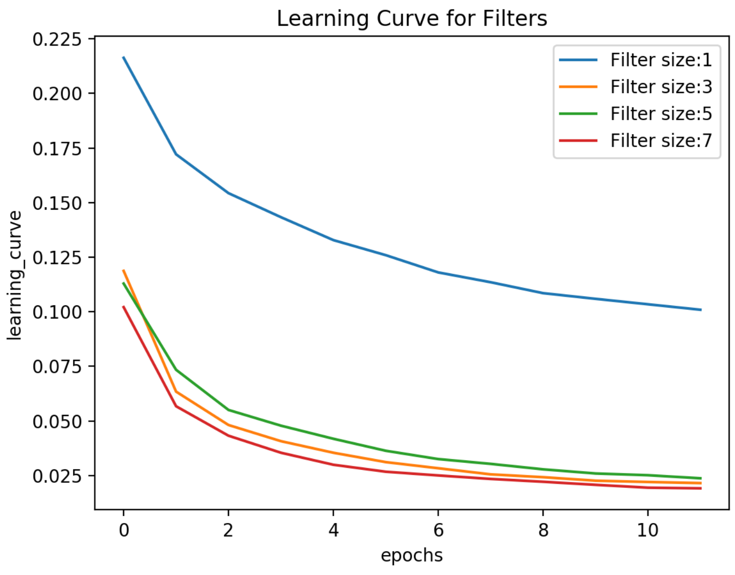

Fig. 3. Loss vs Epochs

From the figure 3, we can make a number of observations.

1. When filter size = 1, the CNN works like a fully connected neural network as there is a weight parameter for each pixel in the image.
2. Increasing the filter size does not have that big a difference on the learning curve, however I expect that with a complicated dataset, the lower filter sizes help in learning more features as we would need more layers to get to the same output image with a lower filter as compared to a larger filter.

Random Search

Finally, we were tasked with running a random search for finding the best configuration of the hyperparameters.
The ranges for various hyperparameters is as follows:-

Hyperparameter	Range	Type
Learning Rate	10^{-4}, 10^{-1}	Float hyperparameter with logarithmic scale
Batch Size	16, 128	Integer hyperparameter with logarithmic scale
Number Of Filters	8, 64	Integer hyperparameter with logarithmic scale
Filter Size	3, 5	Categorical hyperparameter

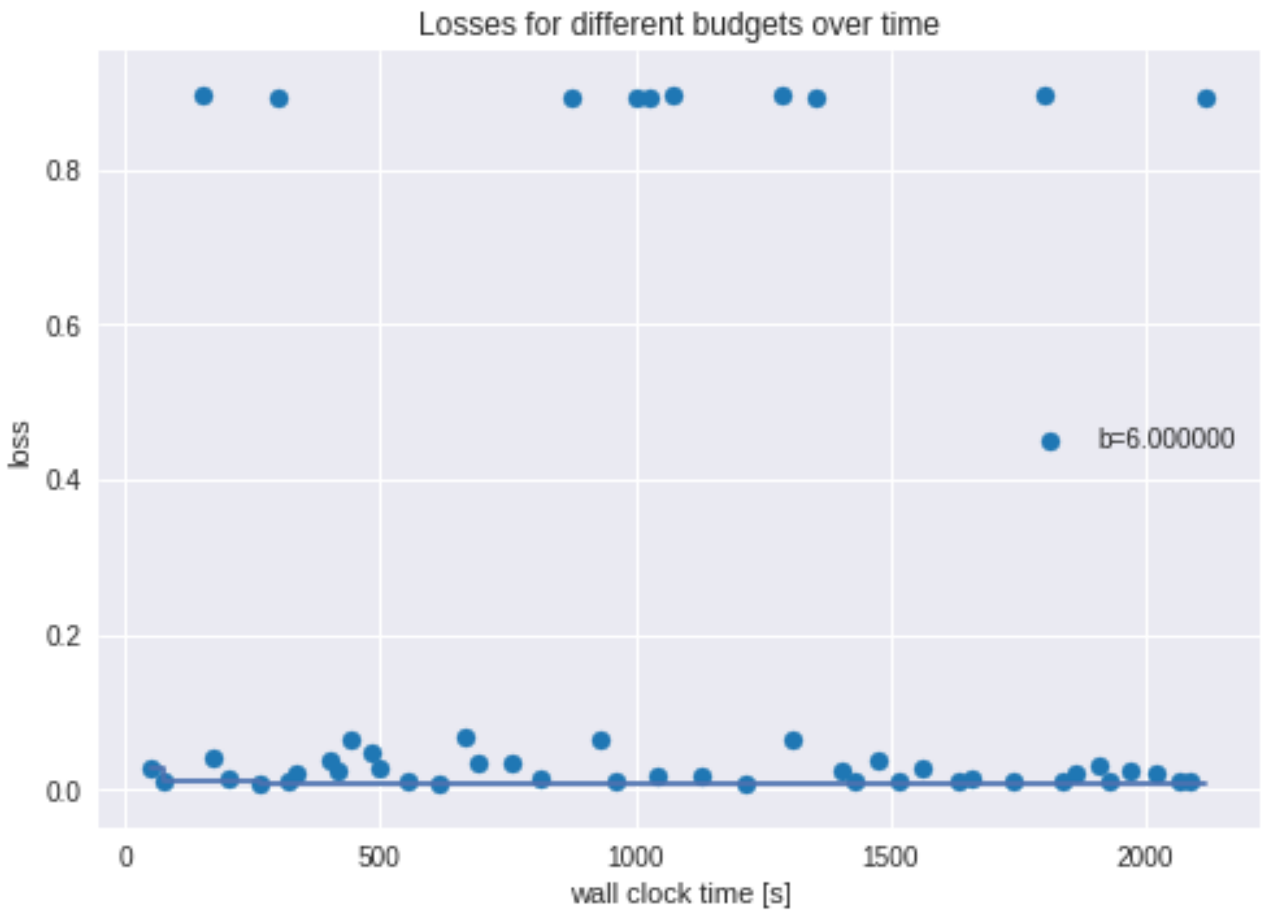

Fig 4. Random Search

The random search found the best configuration to be:-

Hyperparameter	Value
Learning Rate	0.015909904708889904
Batch Size	33
Number Of Filters	57
Filter Size	3

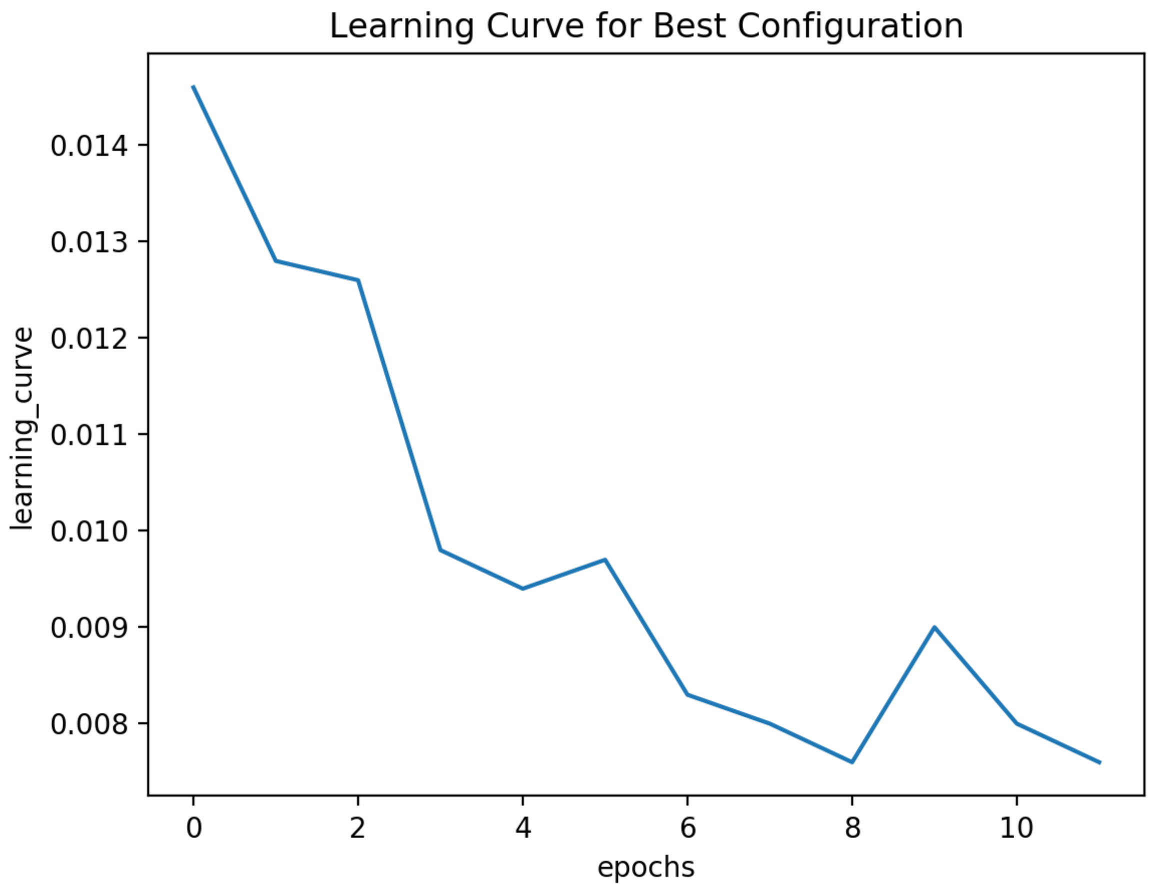

Fig 5. Loss vs Epochs

It can be observed from the curve than after the 6th epoch there is a spike in the loss. This can be attributed to the fact that our random search ran for only 6 epochs therefore the configuration may not be properly optimised.

Problems I faced

1. Firstly, as I had no prior knowledge of tensorflow, I had to solely rely on tutorials to understand it and because of that, I had to rely on trial and error to resolve the errors.
2. Secondly, as the hpbandster package is new, it is tough to understand how it is working and therefore harder to debug. Also, while running on the pool, I faced an error of colormap ‘tab10’ being unavailable and therefore, I had to manually change the visualisation.py for it to use a different colormap.