COL 341

Report – Neural Networks

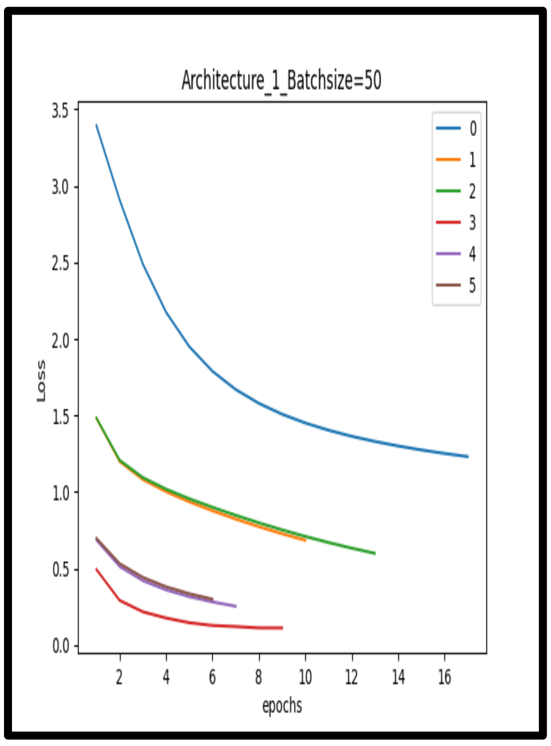
I have tried all the optimizers technique. In part c and d, I have used cross entropy loss function with softmax as activation function in output layer and relu as activation function in hidden layers. In part a and b, I tweaked activation and loss function and I was getting accuracy of 0.86 using tanh activation function, 0.63 using sigmoid activation function and 0.93 using relu activation function for Devanagri data set and cross entropy loss function. Also, using cross entropy loss function was giving better result than MSE. So, used cross entropy loss function and relu activation function in part c and d.

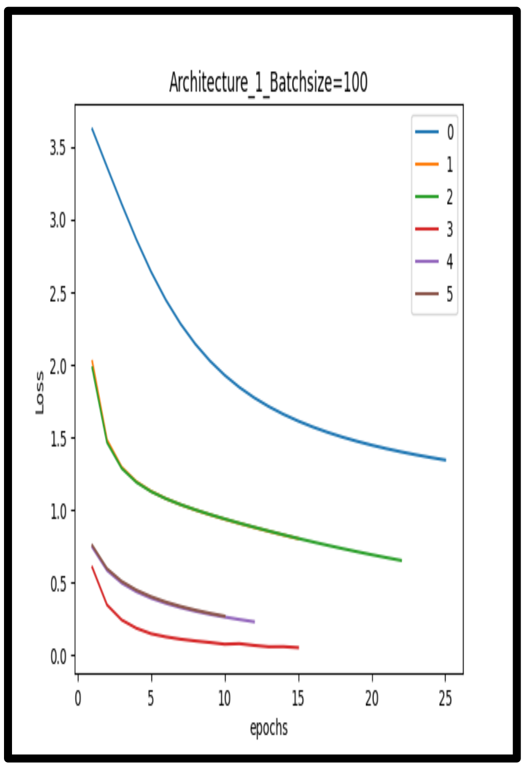
I have created plots of Loss v/s epochs for all these optimizers and have varied batch size. I have used standard values of hyper-parameters. Here, are my standard hyper-parameters for various optimizers:

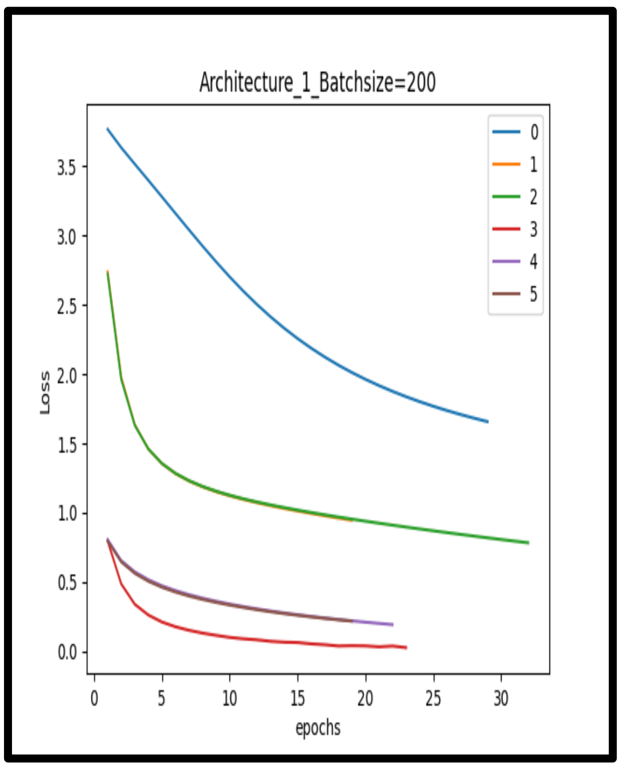
(a) SGD algorithm: Fixed Learning Rate = 0.001.
(b) Momentum and Nesterov: Fixed Learning Rate = 0.001 and $\gamma = 0.9$. I have also tried for adaptive learning rate but it was not giving good results as compared to fixed learning rate.
(c) RMSProp: Fixed Learning Rate = 0.001, $\gamma = 0.9$ and $\in = 10^{-8}$.
(d) Adam and Nadam: Fixed Learning Rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\in = 10^{-8}$.

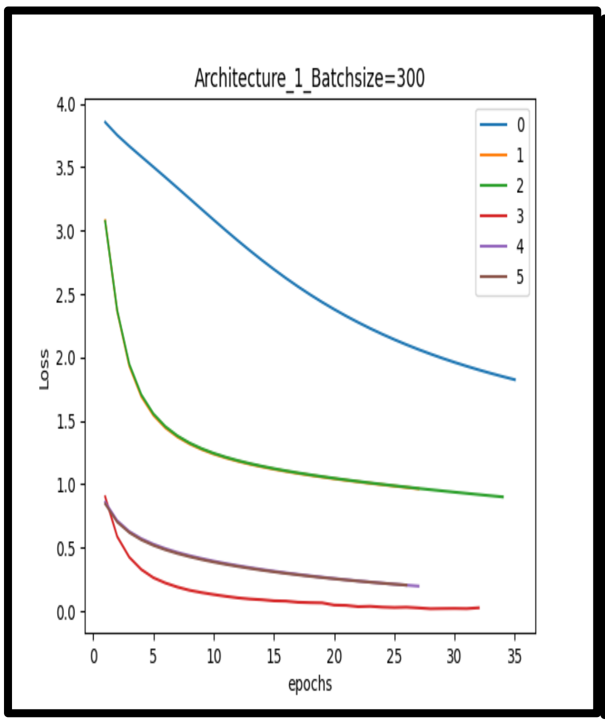
I have plotted Loss v/s epochs for all these optimizers and have varied batch size(In each plot, I have run each optimizer for 300 seconds and stored number of epochs taken by each optimizer and losses in csv files). I have varied batch size = [25,50,100,200,300,500]. Here are my plots for architecture 1 i.e. [256,46]:

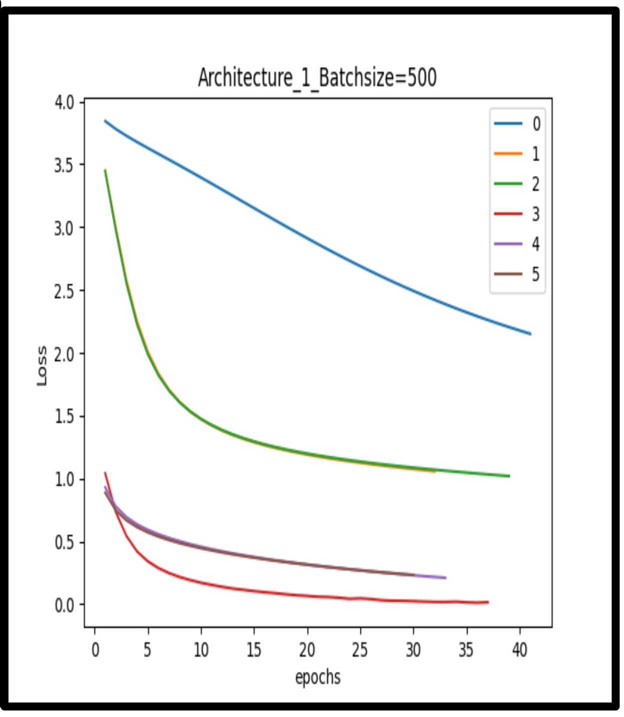
(Here 0-SGD, 1-Momentum, 2- Nesterov, 3-RMSProp, 4-Adam and 5-Nadam)

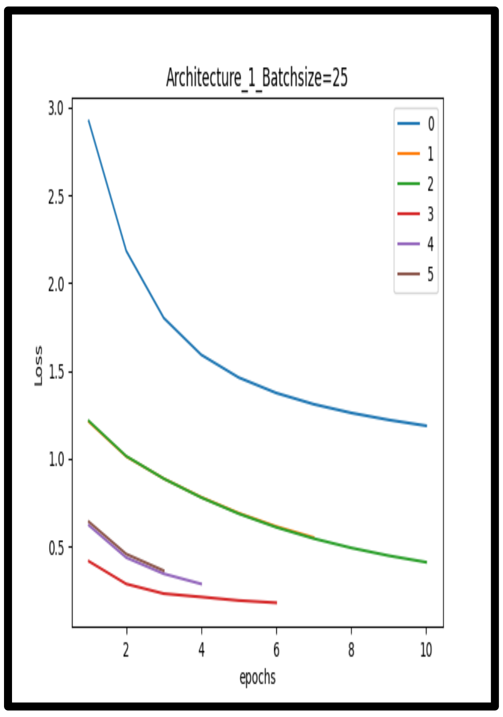




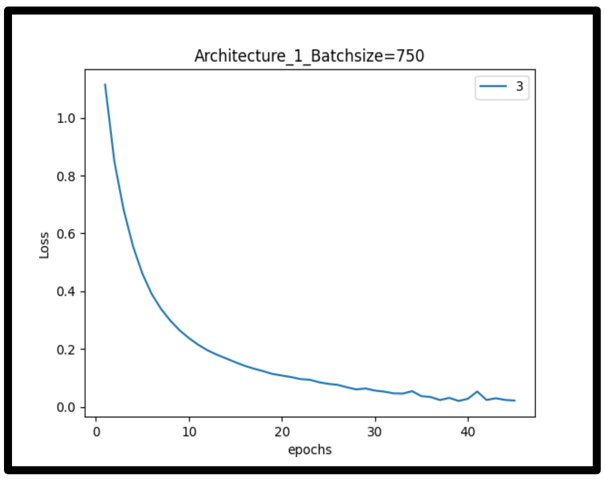


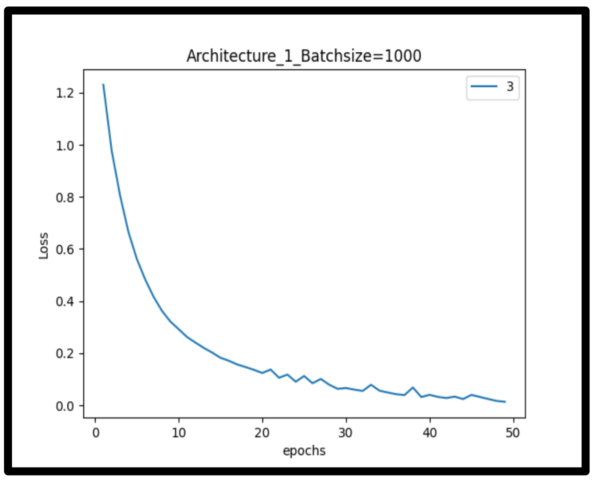




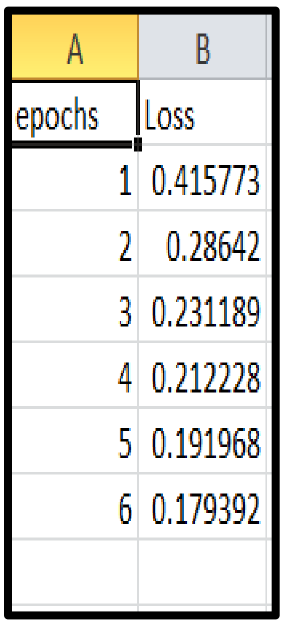


From, the graph, I see that, in each plot RMSProp was giving less loss. So, RMSProp was my best optimizer. Since, I saw that loss was decreasing as I vary batch size and so, I plotted graph of batch sizes 750 and 1000 for RMSProp algorithm. Here, are my plots:



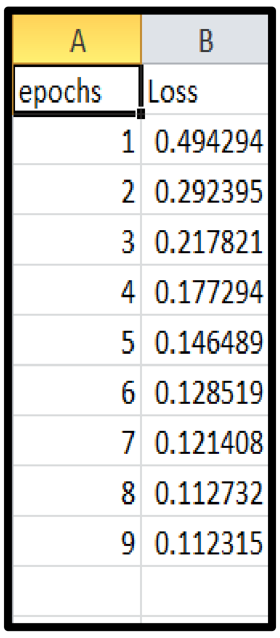


Here are the screenshots of csv files where I stored loss and number of epochs for RMSProp algorithm varying batch sizes:



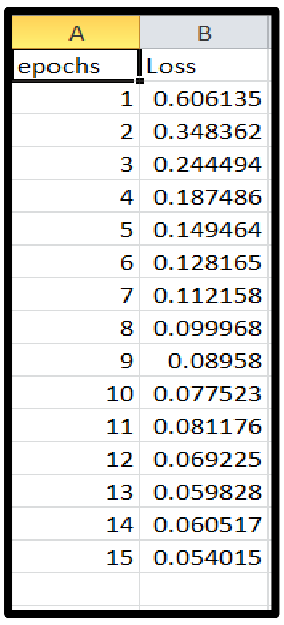
A	B
epochs	Loss
1	0.415773
2	0.28642
3	0.231189
4	0.212228
5	0.191968
6	0.179392

Batch Size = 25



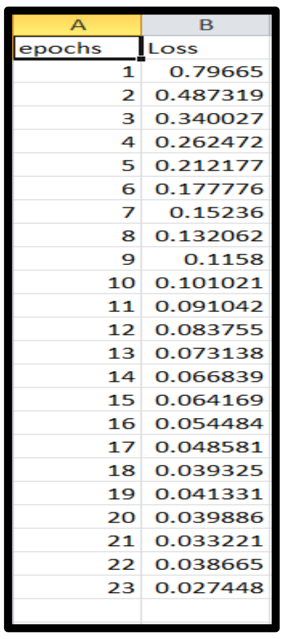
A	B
epochs	Loss
1	0.494294
2	0.292395
3	0.217821
4	0.177294
5	0.146489
6	0.128519
7	0.121408
8	0.112732
9	0.112315

Batch Size = 50



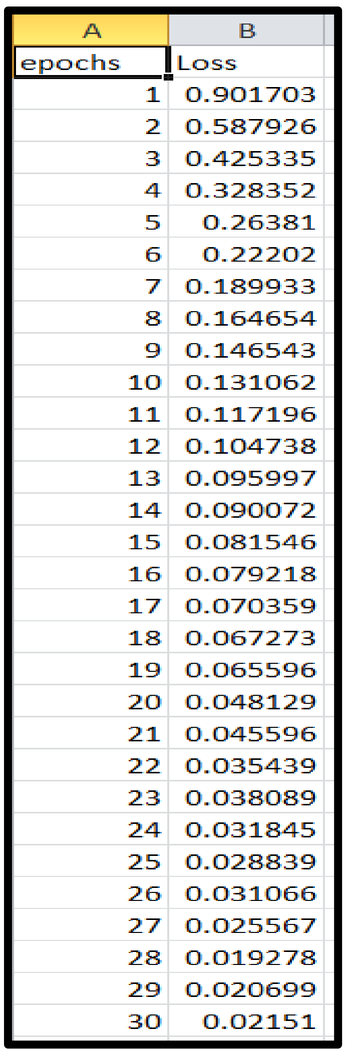
A	B
epochs	Loss
1	0.606135
2	0.348362
3	0.244494
4	0.187486
5	0.149464
6	0.128165
7	0.112158
8	0.099968
9	0.08958
10	0.077523
11	0.081176
12	0.069225
13	0.059828
14	0.060517
15	0.054015

Batch Size = 100



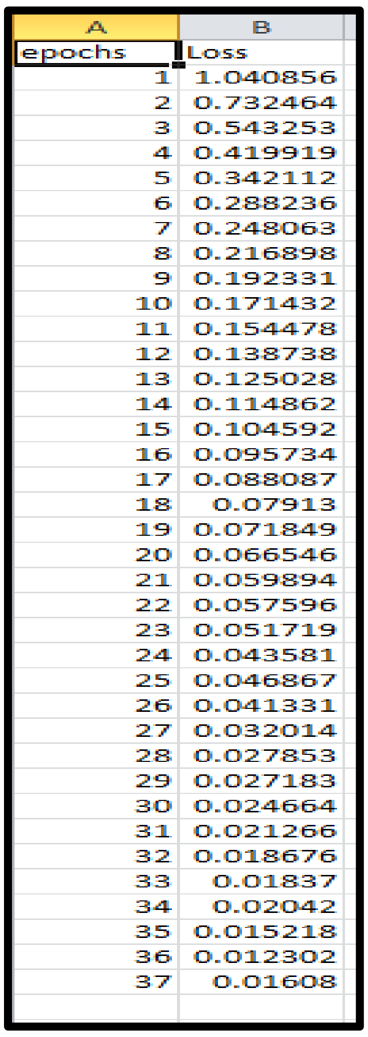
A	B
epochs	Loss
1	0.79665
2	0.487319
3	0.340027
4	0.262472
5	0.212177
6	0.177776
7	0.15236
8	0.132062
9	0.1158
10	0.101021
11	0.091042
12	0.083755
13	0.073138
14	0.066839
15	0.064169
16	0.054484
17	0.048581
18	0.039325
19	0.041331
20	0.039886
21	0.033221
22	0.038665
23	0.027448

Batch Size = 200



A	B
epochs	Loss
1	0.901703
2	0.587926
3	0.425335
4	0.328352
5	0.26381
6	0.22202
7	0.189933
8	0.164654
9	0.146543
10	0.131062
11	0.117196
12	0.104738
13	0.095997
14	0.090072
15	0.081546
16	0.079218
17	0.070359
18	0.067273
19	0.065596
20	0.048129
21	0.045596
22	0.035439
23	0.038089
24	0.031845
25	0.028839
26	0.031066
27	0.025567
28	0.019278
29	0.020699
30	0.02151

Batch Size = 300



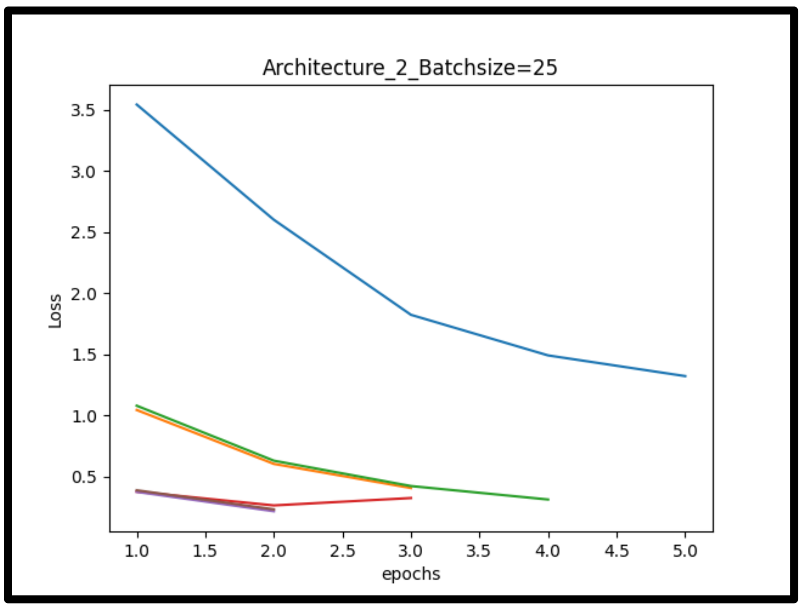
A	B
epochs	Loss
1	1.040856
2	0.732464
3	0.543253
4	0.419919
5	0.342112
6	0.288236
7	0.248063
8	0.216898
9	0.192331
10	0.171432
11	0.154478
12	0.138738
13	0.125028
14	0.114862
15	0.104592
16	0.095734
17	0.088087
18	0.07913
19	0.071849
20	0.066546
21	0.059894
22	0.057596
23	0.051719
24	0.043581
25	0.046867
26	0.041331
27	0.032014
28	0.027853
29	0.027183
30	0.024664
31	0.021266
32	0.018676
33	0.01837
34	0.02042
35	0.015218
36	0.012302
37	0.01608

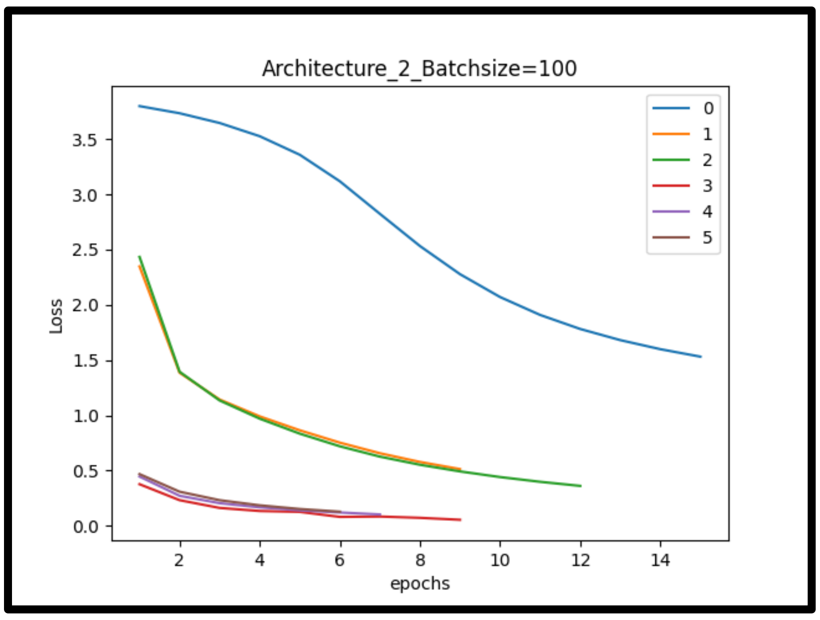
Batch Size = 500

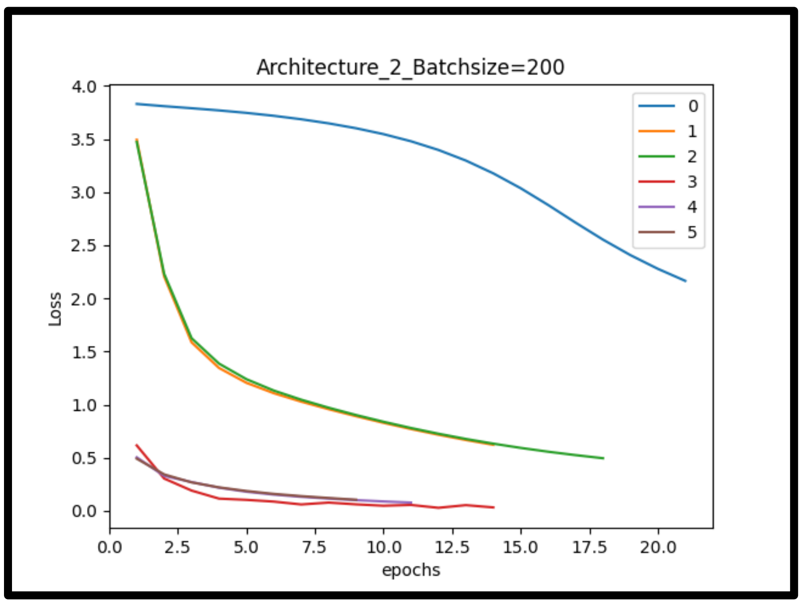
From, the loss values, I found that taking Batch Size = 500 was giving good result and was able to achieve loss = 0.012 using 35 epochs. If I increase batch size to 750,1000. I saw that there was fluctuation in loss values and minimum loss value, I was getting using batch size = 750 is 0.021 and batch size = 1000 is 0.013 which are more than loss value getting using batch size = 500. Hence, in architecture 2, I have not tried batch sizes more than 500.(Here, I have started from epoch = 1 and so, if in csv epoch number = 36 was giving best answer, then it is actually epoch number = 35).

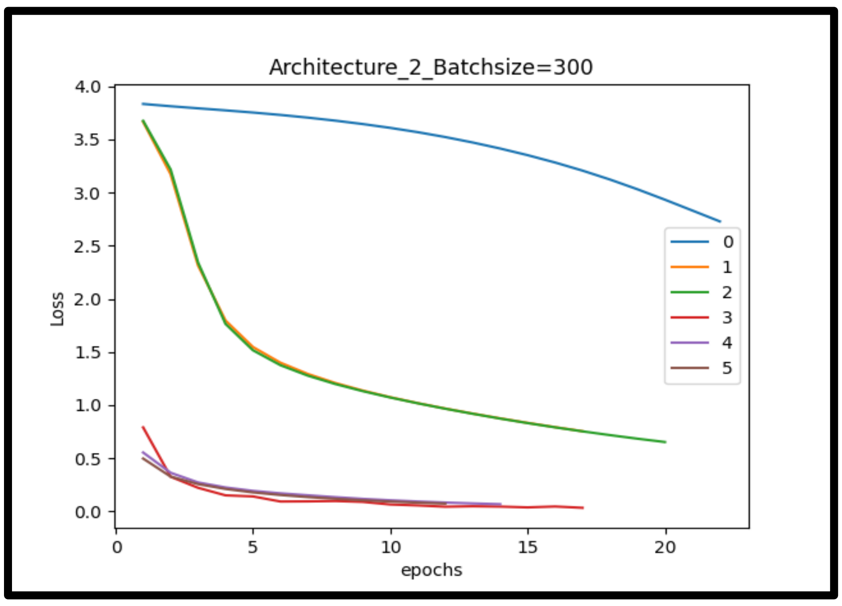
So, best architecture I was getting is to use RMSProp with standard hyperparameters and batch size = 500 and number of epochs = 35. I have initialized the weights using same method as done in part a and b. Used seed value = 1.

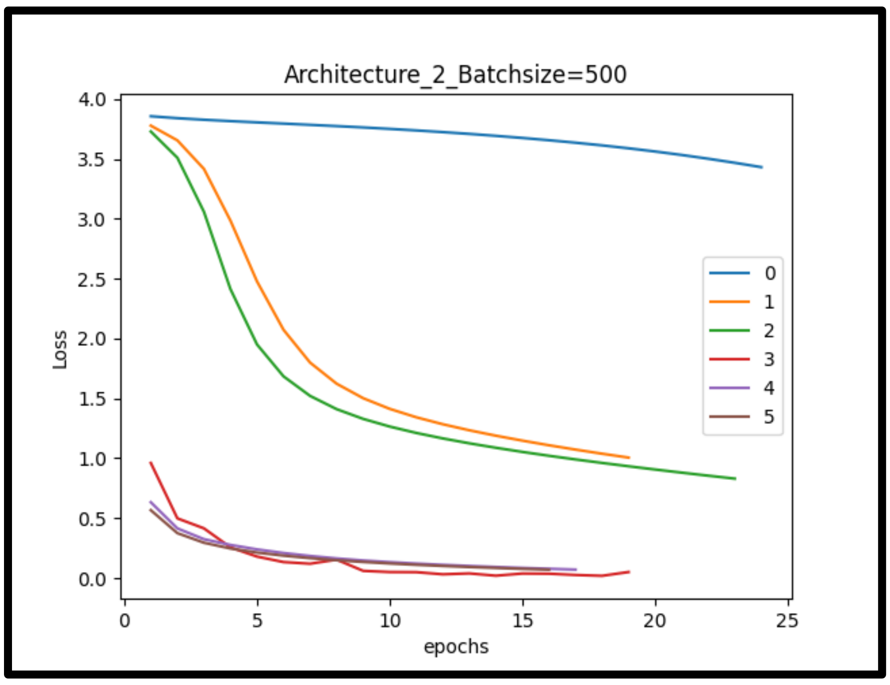
For architecture2 i.e. [512,256,128,64,46] : Here are the plots for various batch sizes and I vary batch size = [25,100,200,300,500]:



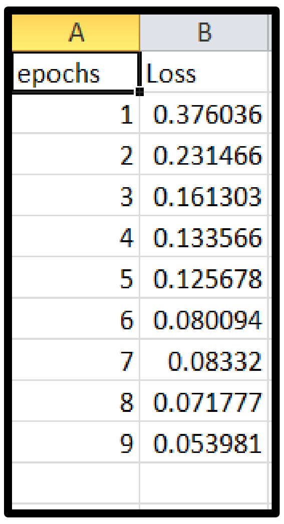






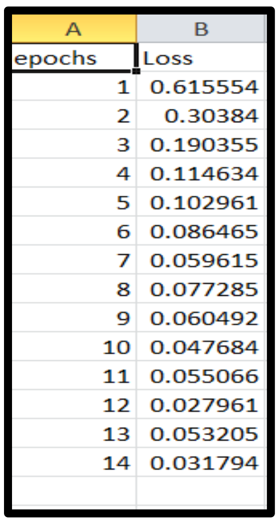


From the graph, using Adam, Nadam and RMSProp was giving good result but RMSProp was giving better result among all of them. Here, are my screenshots of CSV files for RMSProp for various batch sizes:



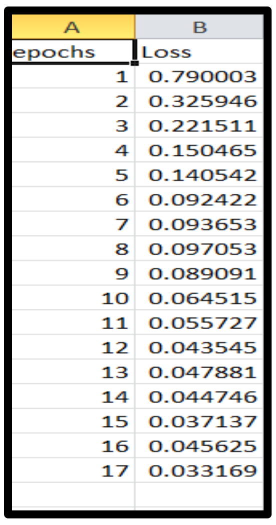
A	B
epochs	Loss
1	0.376036
2	0.231466
3	0.161303
4	0.133566
5	0.125678
6	0.080094
7	0.08332
8	0.071777
9	0.053981

Batch Size = 100



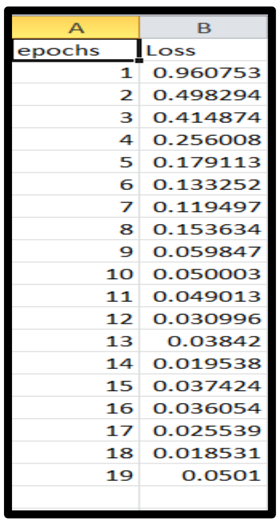
A	B
epochs	Loss
1	0.615554
2	0.30384
3	0.190355
4	0.114634
5	0.102961
6	0.086465
7	0.059615
8	0.077285
9	0.060492
10	0.047684
11	0.055066
12	0.027961
13	0.053205
14	0.031794

Batch Size = 200



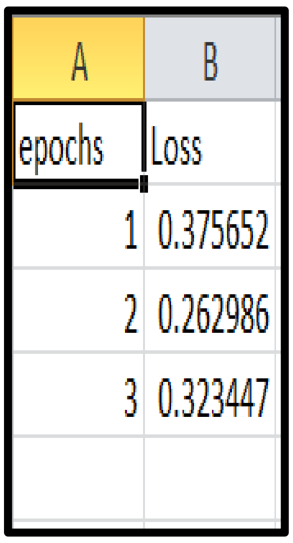
A	B
epochs	Loss
1	0.790003
2	0.325946
3	0.221511
4	0.150465
5	0.140542
6	0.092422
7	0.093653
8	0.097053
9	0.089091
10	0.064515
11	0.055727
12	0.043545
13	0.047881
14	0.044746
15	0.037137
16	0.045625
17	0.033169

Batch Size = 300



A	B
epochs	Loss
1	0.960753
2	0.498294
3	0.414874
4	0.256008
5	0.179113
6	0.133252
7	0.119497
8	0.153634
9	0.059847
10	0.050003
11	0.049013
12	0.030996
13	0.03842
14	0.019538
15	0.037424
16	0.036054
17	0.025539
18	0.018531
19	0.0501

Batch Size = 500

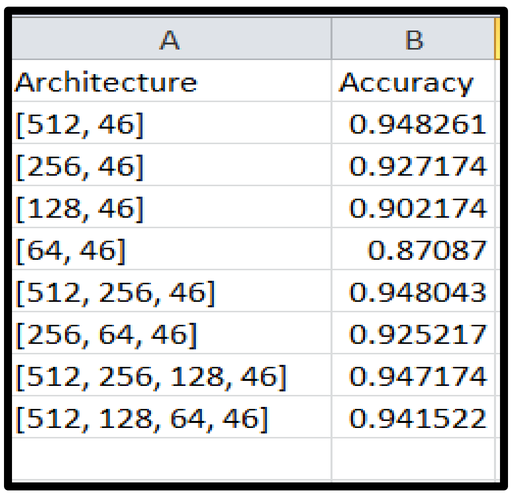


A	B
epochs	Loss
1	0.375652
2	0.262986
3	0.323447

Batch Size = 25

From, the table, I see using batch size = 500 and epochs = 13 was giving minimum loss = 0.019 and so, best architecture will be to use RMSProp with standard hyper-paramteres and batch size = 500, number of epochs = 13.

In part d, I used RMSProp with standard hyperparameters and batch size = 500 and number of epochs = 36. I tried for various architects: [512,46],[256,46],[128,46],[64,46],[512,256,46],[256,64,46],[512,256,128,46]. Here are my prediction accuracy for different architectures :



A	B
Architecture	Accuracy
[512, 46]	0.948261
[256, 46]	0.927174
[128, 46]	0.902174
[64, 46]	0.87087
[512, 256, 46]	0.948043
[256, 64, 46]	0.925217
[512, 256, 128, 46]	0.947174
[512, 128, 64, 46]	0.941522

So, taking architecture [512,46] was giving best prediction accuracy 0.9482 in 300 seconds. So, best architecture for me is [512,46].

Here, I see if I keep number of layers constant and decrease number of neurons in layers, then prediction accuracy decreases. If I increase number of layers than prediction accuracy slightly decreases.