E0 250 Deep Learning : Project 2

HIMANSHU KUMAR (16762)
Mtech(AI)
himanshuks@iisc.ac.in

Tools

- Tensorflow 2.1.0
- Python3 3.6.9
- Keras 2.3.1
- *Keras is using tensorflow as backend.*
- Iam validating my model by using the same test case as given in the keras fashion mnist dataset.

1. **Convolution Neural Network**
   - Accuracy : 0.9319
   - Using the ImageDataGenerator from keras.preprocessing iam rotating my images by 10 degrees, moving their width and height by 0.09 on a scale of 0 to 1, and zooming on the image. Also, adding Gaussian noise for the first 2 stacked layers. Image augmentation prevents overfitting, and reduces the testing error.
   - I have considered 2 architectures with SGD and Adam optimzation on each. First one involved filters stacked as 32, 64, 128. The 2nd one had 32, 32, 64, 64, 128, 128 filters stacked. With dropout, and maxpooling involved latter architecture was performing very well with accuracy >0.9.
   - BatchNormalization normalizes the activation, for each batch. Dropout prevents overfitting, and pooling reduces dimensionality.
   - Iam using selu activation function in intermediate layer to prevent any dead relu's, and also because of it's fast learning.
   - 50 epochs, batch=32, optimizer = adam
     For the epochs >40, I got >90% validation accuracy.
     I have used a kernel size of 3x3 as the images are of 28x28.
   - I have acheived better performance with Adam as compared to SGD, maybe as it combines momentum to RMSProp and SGD with momentum. I have used the default learning rate.

Confusion Matrix :

- The network struggles with classifying class 0, and confuses it with class 6 mostly. Also, for the class 6, network misclassifies it with 0,2,3,4 class.
- For the rest classes, network performs well.

Actual/Predicted	0	1	2	3	4	5	6	7	8	9
0	843	0	19	7	2	0	126	0	3	0
1	0	992	0	7	0	0	0	0	1	0
2	10	0	906	9	35	0	40	0	0	0
3	6	4	8	927	18	0	36	0	1	0
4	0	0	23	23	910	0	44	0	0	0
5	0	0	0	0	0	992	0	6	0	2
6	57	0	37	14	59	1	828	0	4	0
7	0	0	0	0	0	13	0	975	0	12
8	0	1	1	3	0	0	1	0	994	0
9	0	0	0	0	0	6	0	41	1	952

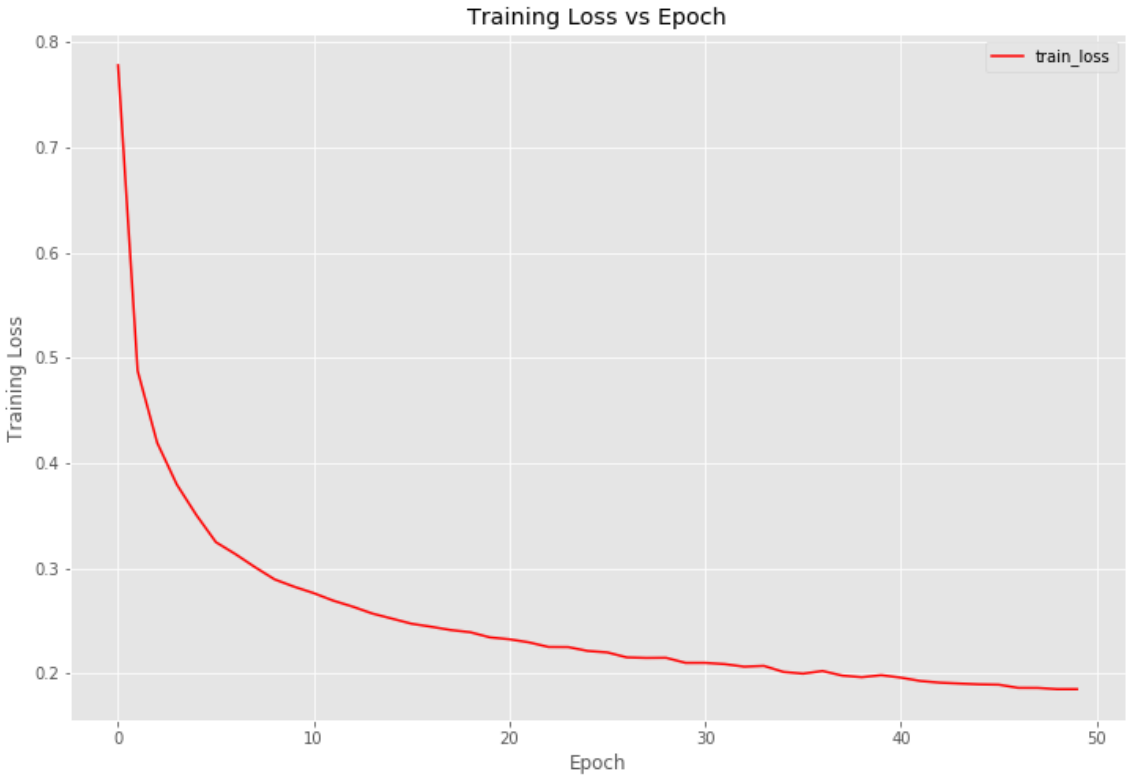


Figure 1: Training loss vs Epochs

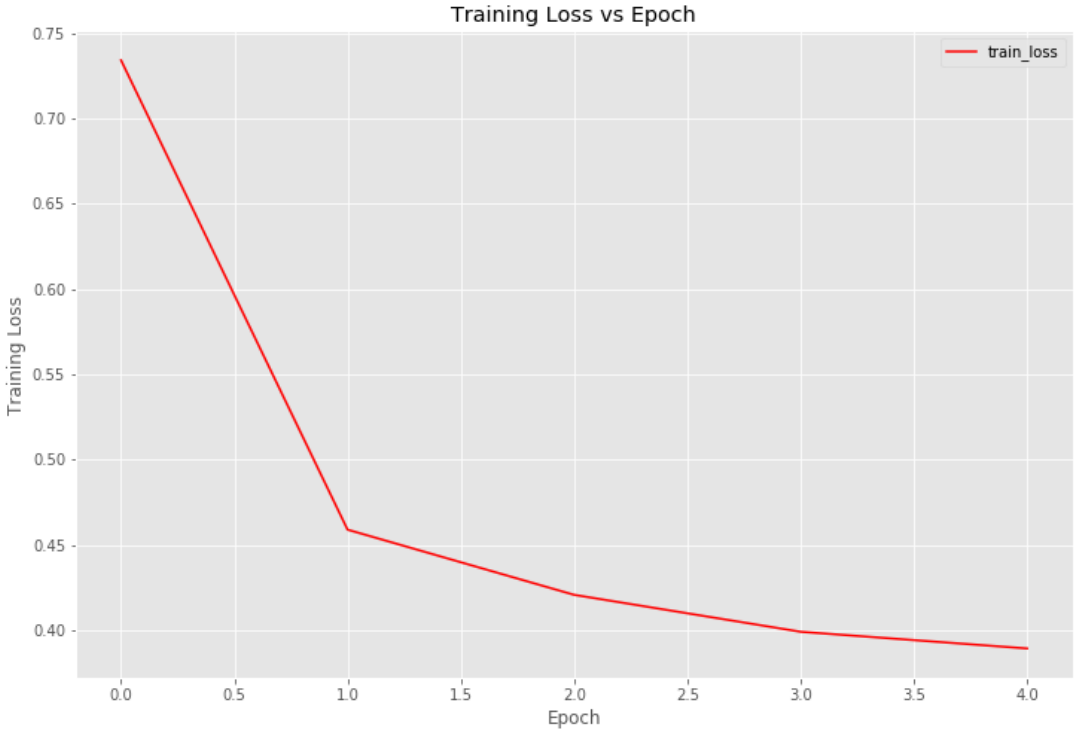
2. **Multi-Layer Neural Network**

- Accuracy : 0.8465
- I haven't used any image augmentation with MLP, as the performance was better on the original data. Also, adding any Gaussian noise to image resulted in less training accuracy.
- I tried with 2 models, initially having 256 filters while other model having 512 filters. Model with 512 filters performed better, as more filters are present in deeper layers.
- BatchNormalization normalizes the activation, for each batch. Dropout prevents overfitting, and genralizes in a fully connected NN.
- I used selu activation function for 2nd layer to prevent any dead relu's, and also because of it's fast learning.
- 5 epochs, batch=128, optimizer = RMSprop
  After epoch 4, there was less observable validation accuracy difference.
  I have used the default learning rate, as it's optimal.

Confusion Matrix :

- The network has a deep problem classifying classes 0, 2, 4, 6. Also, class 7.
- For the rest classes, network performs well.

Actual/Predicted	0	1	2	3	4	5	6	7	8	9
0	737	5	14	81	3	10	140	0	10	0
1	3	948	3	39	4	0	1	0	2	0
2	13	0	725	12	120	7	113	1	9	0
3	12	5	9	933	11	2	25	0	3	0
4	0	2	80	59	743	10	100	0	6	0
5	0	0	0	2	0	956	0	19	1	22
6	88	2	86	74	77	19	636	0	18	0
7	0	0	0	0	0	47	0	892	1	60
8	0	2	4	12	2	21	17	8	933	1
9	0	0	0	0	0	14	0	23	1	962


Training Loss vs Epoch
train_loss
Training Loss
Epoch
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0