DeepLearning Assignemnt -2, 2020
Ramkiran – 17545

The assignment is about constructing a deep network on fashion MNIST Data using
1. Dense Neural Network
2. Convolution Neural Network

Dense Neural Network

Dense neural network was constructed in pytorch frame work provided in Github. There are several hyper parameters to be modified to find the best possible network. Some of the hyper parameters that can be used are

1. Number of hidden layers.
2. Number of neurons per layer
3. Optimizer Choice & Learning Rate

Also to increase the robustness of the network data augmentation by adding random noise to the image data is done during training. Drop out layers are also added and later removed as the effect on the network accuracy is less sensitive in this case. A maximum hidden layers of 3 only is used to avoid the effects of vanishing gradients. Relu activation functions are used in the hidden layers to avoid pit-falls/exploding gradients. Validation and training sets are formed in 20% to 80% ratio while training.

Number of hidden layers:
Initially only one hidden layer is used to form the network. The number of neurons required to represent the neural network model if of the order 2^n (from universal approximation theorem) for 1-hidden layer network. As the depth of the network increases the no.of neurons required decreases considerably and hence a deeper network is used in the assignment. Too much depth would cause vanishing gradient problems and hence maximum depth was limited to 3 in this assignment.

Number of neurons per layer:
To avoid over-fitting/ under-fitting model was tested for validation loss and test loss for various choices of neuron count in each layer. A grid based search is done (grid on the parameter size) to find the hyper parameters (number of neurons per layer) with minimum loss and minimum difference between validation and train loss, for 20 epochs. Various best parameters sets are extracted from the grid search and three such samples are tabulated below.

Results of validation and train set accuracy in grid based search are also provided. Final accuracy on the test set is also provided in the results.

Optimizer Choice & Learning Rate:
Adam optimizer is selected for the gradient descent method as it has properties of momentum and adaptive learning rate (like RMS prop). To find learning rate learning, several learning rates in logrthematic scale were tried (i.e. [0.1 0.01 0.001 0.0001]) and picked a learning rate (initial - 0.001) that best minimizes the loss function.
Network:
Some Selected Best Network - Performance (50 epochs)
n1=320,n2=96,n3=32 | Test Accuracy: 0.8992999792098999 Test Loss: 0.29948028922080994
n1=384,n2=128,n3=32 | Test Accuracy: 0.9039000272750854 Test Loss: 0.2879541516304016
n1=384,n2=128,n3=64) | Test Accuracy: 0.8995000123977661 Test Loss: 0.300533652305603
where n1,n2,n3 are the the number of neurons in layers 1,2,3 respectively

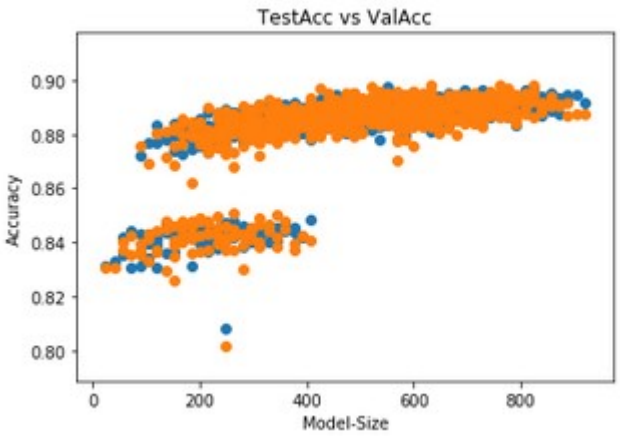


Figure showing validation vs Train Accuracy for different model sizes

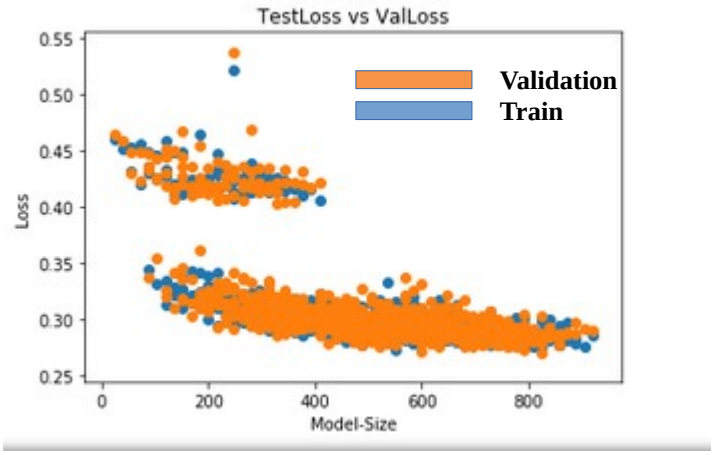
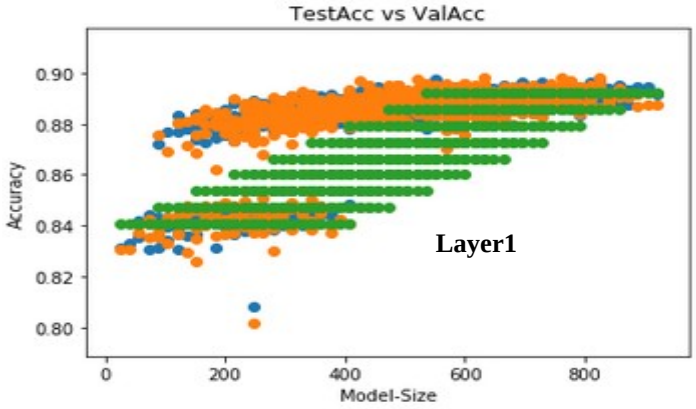
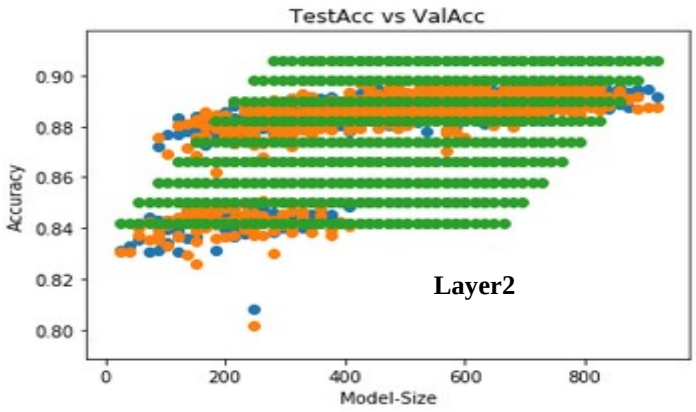


Figure showing validation vs Train loss for different model sizes





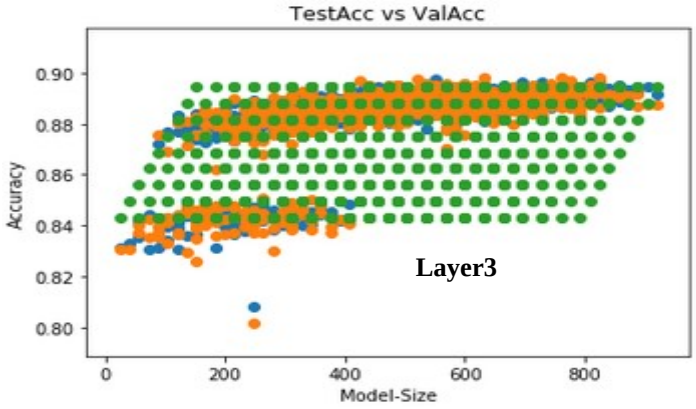


Figure showing Parameters vs Model Accuracy. Here parameters are the number of neurons per layer

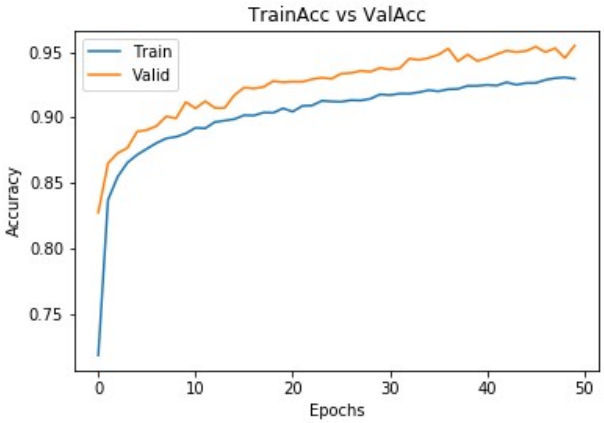


Figure showing validation vs Train Accuracy for n1=320,n2=96,n3=32

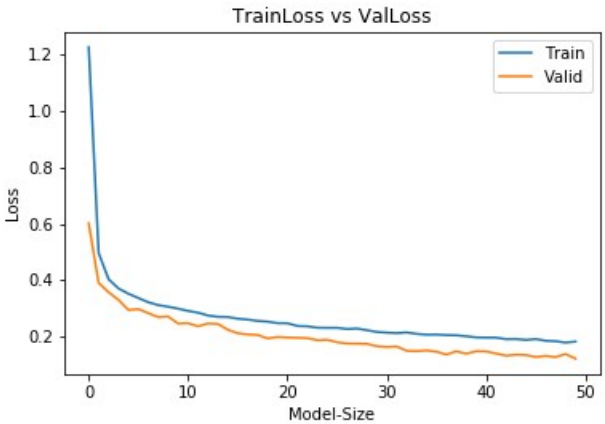


Figure showing validation vs Train Loss for n1=320,n2=96,n3=32

Conclusion:

After selecting the model hyper-parameter for Dense network an accuracy ~ 90% is achieved with DNN.

Convolution Neural Network

The second part of the assignment is to implement a CNN to classify the fashion MNIST data. In light of system requirements (RAM Full happened) for implementation in pytorch, final version of CNN was implemented in tensorflow (Google colab).

Selection of hyper parameters is done similar to DNN case. But the number of elements in the grid are reduced as the time to tune a single CNN is very high compared to the DNN counterpart.

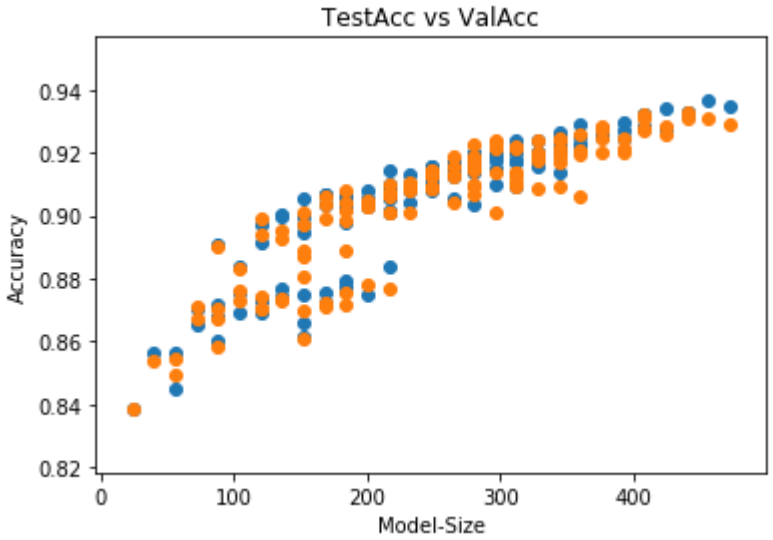


Figure showing validation vs Train Accuracy for different model sizes. Number of grid points tried are less as the time taken for convergence in each grid point is considerably high compared to its DNN counterpart.

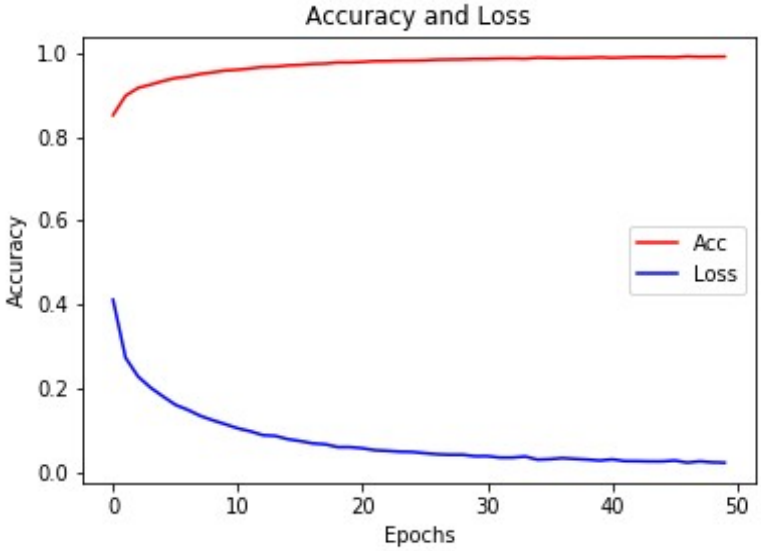


Figure showing Accuracy and loss for one of the best cases in the CNN based design. A train accuracy of 98% was observed and test accuracy of ~ 90% was achieved

Conclusion:

After carefully selecting the model hyper parameter for Convolution network an accuracy ~ 90% is achieved with DNN

Model Summary

DNN

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 320]         251,200
       BatchNorm1d-2                  [-1, 320]             640
            Linear-3                   [-1, 96]          30,816
       BatchNorm1d-4                   [-1, 96]             192
            Linear-5                   [-1, 32]           3,104
            Linear-6                   [-1, 10]             330
================================================================
Total params: 286,282
Trainable params: 286,282
Non-trainable params: 0
```

CNN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 13, 13, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 5, 5, 64)          256
_________________________________________________________________
flatten (Flatten)            (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 16)                25616
_________________________________________________________________
dense_1 (Dense)              (None, 10)                170
=================================================================
Total params: 44,986
Trainable params: 44,794
Non-trainable params: 192
```

Conclusion:

CNN could achieve same accuracy as DNN with less number of parameters