CS7GV1 Computer Vision Assignment P2 Report

Anjoe Anand Jacob

***Abstract*— This report details the deep learning techniques applied to Fashion-MNIST dataset and their results as part of Assignment P2 in CS7GV1 Computer Vision module.**

I. Introduction

This assignment uses Python with PyTorch to design and train a model to recognize clothes from Fashion-MNIST dataset.[1].

II. Fashion-MNIST dataset

The dataset contains 70000 grayscale images of size 28x28. This has been split to 60000 for training and 10000 for testing. The [2] dataset comprises of 10 different apparel types (Fig -) This is a popular dataset and is more challenging than MNIST.

III. Baseline

The baseline was established using the code used during the lab sessions. Fashion-MNIST data was downloaded by replacing the existing MNIST function call in the code. The required libraries are imported, followed by declaration of hyperparameters. The program makes use of GPU to increase the computation throughput using CUDA. In a system without a GPU, the code switches execution to CPU. Up next is the download of training and testing Fashion-MNIST dataset respectively. The model is created and is followed by instantiating the optimizer. This is followed by training the model based on the number of epochs set. Lastly, the graphs showing train and test error, and loss are plotted.

IV. Network Design and Motivation

I started off simple with just 2 fully connected networks. This network was trained with 10 epochs, a batch size of 64 and a learning rate of 0.01. The optimizer used is Stochastic Gradient Descent (SGD) which updates the parameters for training examples one by one [2]. An accuracy of 85.5% was obtained using this basic settings. I started looking into ways that can improve the accuracy and for better design patterns[3]. I added a single Convolutional Neural Network (CNN) to the network. With other parameters unchanged, the accuracy was increased to 89.12%. Nitish Srivastava et al. [4] suggested dropouts that will randomly drop units from neural network during training. This helps preventing too much co-adaptation. Furthermore, the concept of maxpooling was also implemented which helps in down-sampling and thereby making the training process faster [5].

ReLU activation functions were used after every layer which helps in determining the output like a yes or no [6].

$$R(z) = max(z, 0)$$

Other activation functions like LeakyReLU and tanh were also tried, but ReLU performed better.

A little more research on the topic resulted in finding out about Inception architecture where multiple filters are applied and later concatenated [6]. The below network was implemented and an accuracy of 90.53% was obtained.

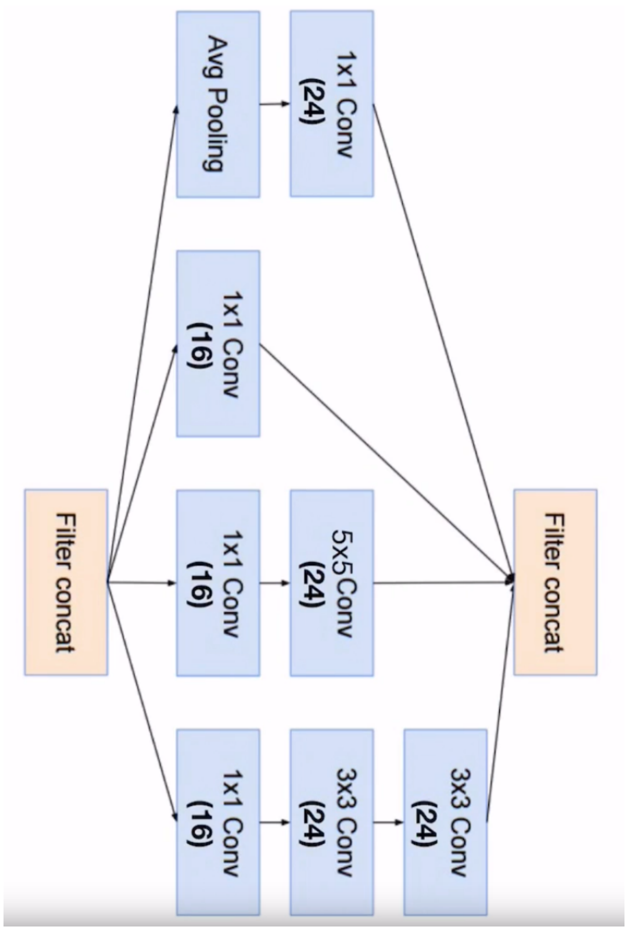


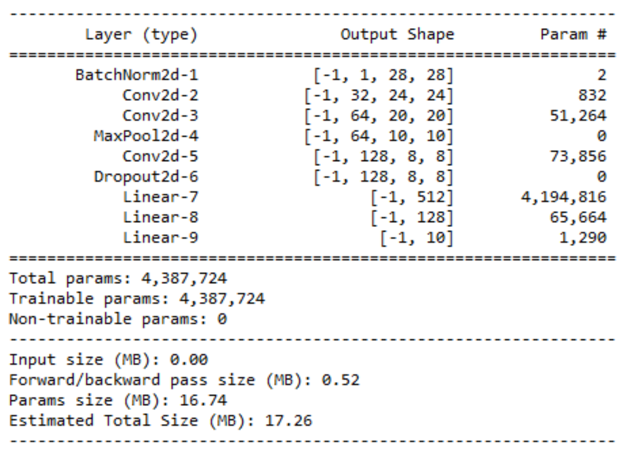
Fig. 1. Inception Naive [6]

More complicated networks like Inception v2, Inception v3, and Inception v4 were not implemented.

Further reading lead to the idea of combining multiple CNNs and fully connected networks with a combination of batch normalization, max-pooling and dropouts. The results so far and the idea of combination motivated me to test different combinations of networks. The final network is discussed in detail in section V.

V. Combination of Networks

The architecture uses 3 CNNs and 3 fully connected networks. The input layer was subjected to Batch Normalization and then fed to CNN layer 1 with 1 input channel and 32 output channels with a kernel size of 5 with default stride and padding. ReLU activation function is applied to the output and its output is fed to CNN layer 2 with 32 input channels and 64 output channels. Its output is fed to a maxpool function and subsequently connected to ReLU activation. This is then connected to the 3rd CNN with 64 input and 128 output channels respectively. This is then connected to ReLU function and a dropout of 50% is done. This output is connected through 2 pairs of fully connected network and ReLU back-to-back and then given to the 3rd fully connected network from where it passes through the log softmax function. The optimizer used is SGD and loss function used in nll loss.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
       BatchNorm2d-1            [-1, 1, 28, 28]               2
            Conv2d-2           [-1, 32, 24, 24]             832
            Conv2d-3           [-1, 64, 20, 20]          51,264
         MaxPool2d-4           [-1, 64, 10, 10]               0
            Conv2d-5            [-1, 128, 8, 8]          73,856
         Dropout2d-6            [-1, 128, 8, 8]               0
            Linear-7                  [-1, 512]       4,194,816
            Linear-8                  [-1, 128]          65,664
            Linear-9                   [-1, 10]           1,290
================================================================
Total params: 4,387,724
Trainable params: 4,387,724
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.52
Params size (MB): 16.74
Estimated Total Size (MB): 17.26
----------------------------------------------------------------
```

Fig. 2. Summary of the model

VI. Results

The network performed with 92.52 % accuracy on test data with the following hyper parameters.

epochs = 30
batch size = 64
learning rate = 0.01
momentum = 0.9
log interval = 20

The model has been saved as *3c3f.pt*. Results of the predictions have been saved into *predictions.txt*. These files are are available at `https://drive.google.com/open?id=1xfbsvCFy0L-Msy686xx7K_GtLjBtNoVn`

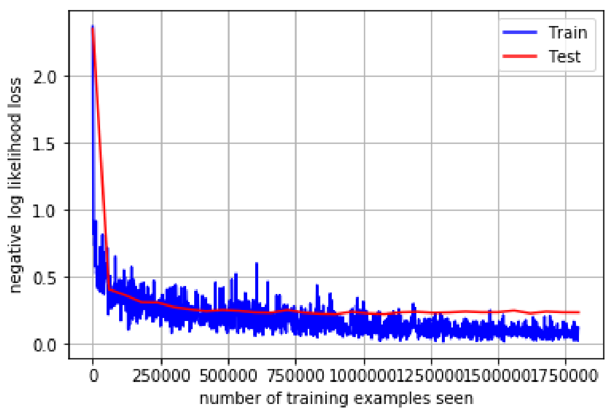


Fig. 3. Graph showing loss

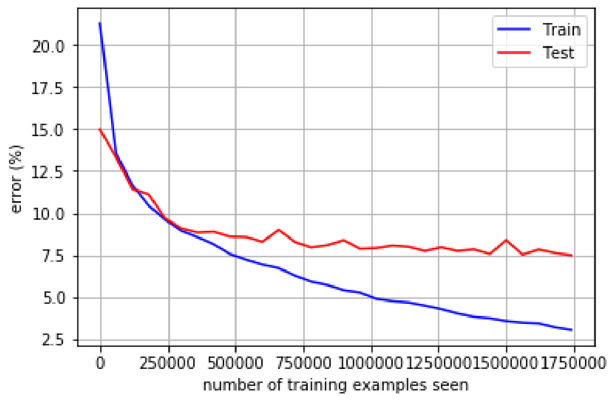


Fig. 4. Graph showing error

References

[1] J. C.-M. . J. F.-V. J. L. Pech-Pacheco G. Cristobal, "Diatom autofocusing in brightfield microscopy: a comparative study," 2000.
[2] "Sgd - https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0,"
[3] "https://datascience.stackexchange.com/questions/20222/how-to-decide-neural-network-architecture,"
[4] "Dropout: A simple way to prevent neural networks from overfitting - https://www.cs.toronto.edu/ hinton/absps/jmlrdropout.pdf,"
[5] A. M. Dominik Scherer and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition - http://ais.uni-bonn.de/papers/icann2010$_maxpool.pdf,''$
[6] "Inception naive - https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202,"

Appendix

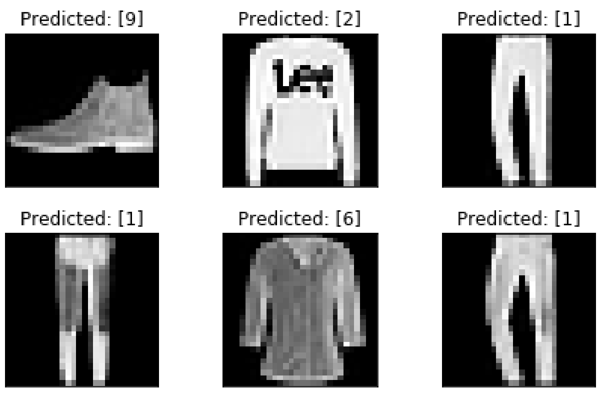
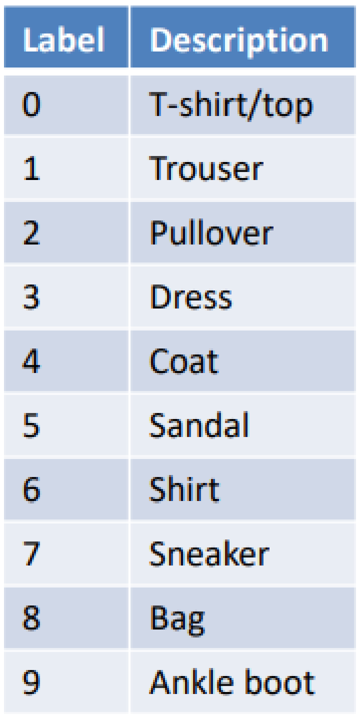


Fig. 5. Results of Prediction)



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Fig. 6. Results of Prediction