CSA 250 : Deep Learning Project II Report

Akhilesh Yadv
Sr.No 17410

February 21, 2020

1 Introduciton

This project is to implement neural network and convolutional neural network for the task of classifi-cation. The classification task will be that of recognizing an image(FASHION MNIST) and identify it as one of ten classes. File Structure is given below model testing *python main.py*

```
Assignment_2
├─ fminstData
│  └─ fmnistData.py //data_generater
├─ images
│  └─ *store_image.png//stored images
├─ models
│  ├─ saved_model_cnn_best.h5 //cnn model weight
│  └─ saved_model_mlp_best.h5 //Multilayer model weight
├─ ploting
│  └─ myplot.py
├─ main.py
├─ model.py
├─ training_conv_net.py
├─ training_mlp.py
├─ multi-layer-net.txt
├─ convolution-neural-net.txt
└─ Deep_Learning_Report_2.pdf
```

Data is used by keras load_data API. I implement 2 model of the data classification by using multilayer perceptron and by using CNN input None output 2 file generated Multilayer CNN

Tools:

Python 3.7 Tensorflow 2.0.0 with tf.keras Numpy Scikit-learn RUN CODE: As given in the instruction

Final output:

	Accuracy		
Model	Train	Validation	Test
Multilayer	95.03	92.8	91.97
CNN	96.03	94.12	93.44

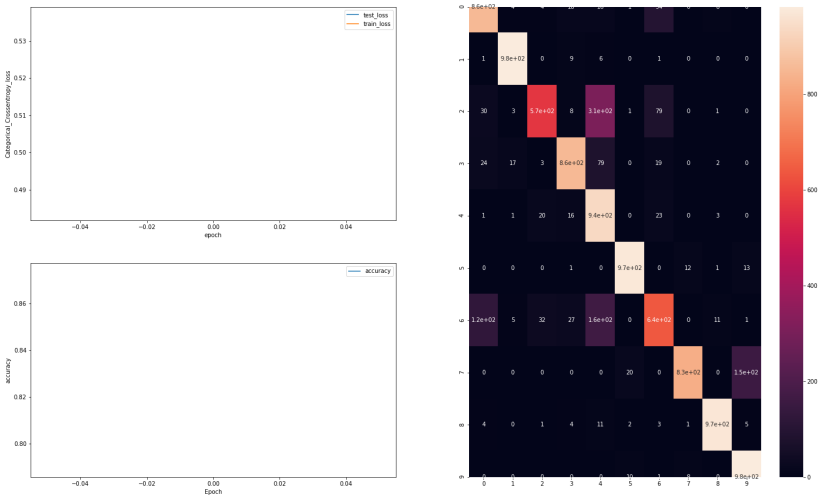

Figure 1: CNN outpur

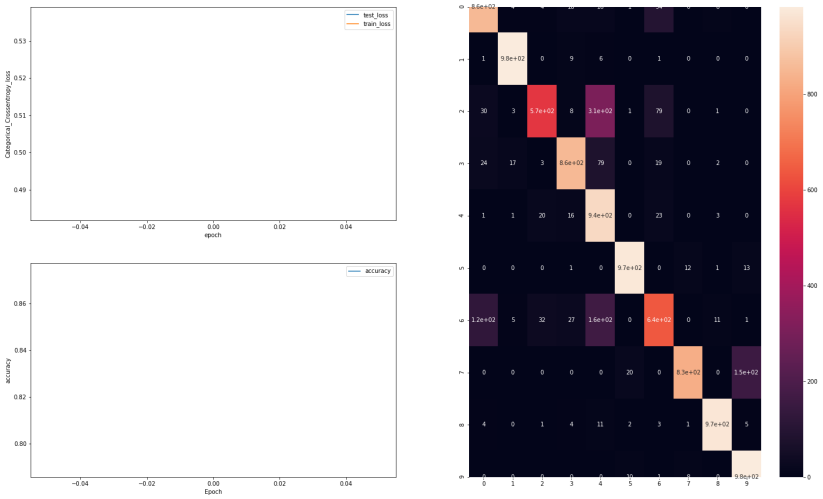

Figure 2: MLP output

2 Multilayer Perceptorn

2.1 Epoch and Batch Size

Initial Batch size I take 1024 after 10 epoch I decrease the Batch size to 128 till 20 epoch then train 50 epoch at 64 batch size.

Reason:
Initialy take a large batch size so that it *decrease faster* then decrese the batch size so that computaion cost can reduce and also add some stochastic noise to data to overcome the problem of overfitting.

2.2 Learning rate and Optimization algo

Try defferent type of optimization and lr. best finding is initialy take large $lr = 0.1$ then decrease the $\frac{lr}{2}$ after every 10 epoch till 30 epoch then constant $lr = 0.002$

Reason:
Initial lr is more to fasten the training process then decrease the lr same time batch size to prevent from oscilation. For the same (other parameter like optimization(RMSprop),Momentom(0), learning rate(0.02), bs(64))

2.3 Neural Network Arch

Dept of neural network & Number of neurons in a layer: Tried several architechure

Finding By increasing the number of layer training process is decreases and also after 6-7 layer model became overfited and training accuracy goes to .99 but as the same time validation and test accuracy stuck at $.70 - .80$ so it does not help to increase the number of layer to get more accuracy. At the same case with number of neurons in a layer it does not help too much to increase the accuracy.

Intuitive reason when i checked for the weight of neurons in large number of neurons the most of the neurons goes to zero and some weight is too high. may be it just copying the data from form fist layer to next layer.

2.4 Batch Normalization

Batch normalization is done by reducing the feature by 255 because in image max posible val is 255 so it convert the data in range of 0–1.

It affect more when I tried with without normalisation i get an accuracy about .70 but after duing the normalisation it increases.

2.5 Noise

Noise 2 types of noise is added

$$Gausiannoise \sim \mathcal{N}(0,\ .05)\,.$$

$$Salt\ and\ Pepper\ noise\ with\ prob(.1)$$

3 CNN

Similar setup as the above for multilayer except

3.1 Network Architecture

I tried different parameter like

kernel $\begin{bmatrix} 3*3*64 & 3*3*32 \end{bmatrix}$

kernel $\begin{bmatrix} 5*5*64 & 5*5*32 \end{bmatrix}$

Maxpolling $2 * 2$

Avgpolling $2 * 2$

Effect of kernel By increasing kernel shape from $3 - 5$ accuracy decrease for the same architecture and going from $5 - 7$ it's about .67 may be due to small image size. But by increasing the network depth it is unpredictable.

Effect of Polling It more matter where and how many you used polling layer because for larger number of polling layer decrease the feature shape and netork may not hold enough featur to pass in dense layer. My final architecture I put *2–Conv followed by 1 polling and 1 dropout(0.25) this repeat for 2 times.*

Reason I found that when applying the *Conv-Polling-Conv* then it is not affecting more may be reason is sandwiching polling layer between Conv layer decrease the feature size before extracting non-local feature by next layer.