Megha Sinha – 71189549

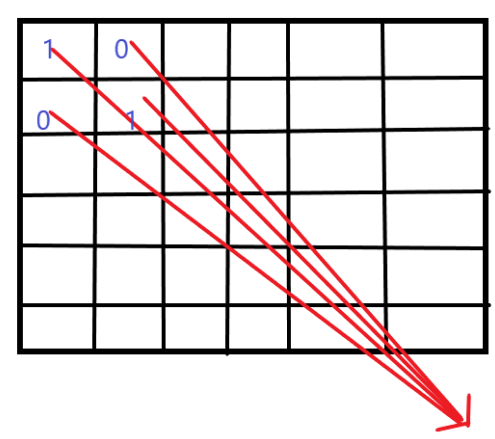
Siddhartha - 58651262

Big DaTA – Assignment3

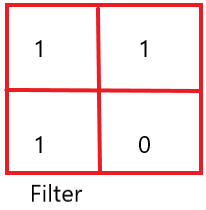
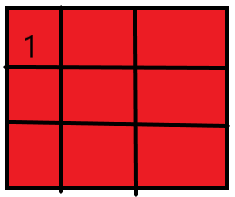
**Question 1(a)**

**Question 1(b):**

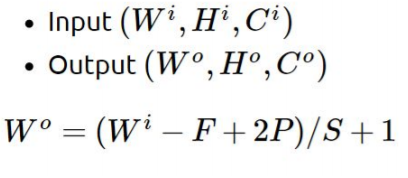
Input



Applying Filter which gives the below output

  Output

**Output Size:**



= ((6-2)/2) +1 = 3

**Parameter:**



= (2\* 2 \* 1 + 1) \*1 = 5

**Question 2:**

For this part of question, we have chosen Fashion-MNIST dataset. This dataset consists of 60000 observations of images of 10 fashion categories, along with a test set of 10,000 images. Each image is grey-scale and of size 28\*28 having the below class labels-

T-shirt/top

Trouser

Pullover

Dress

Coat

Sandal

Shirt

Sneaker

Bag

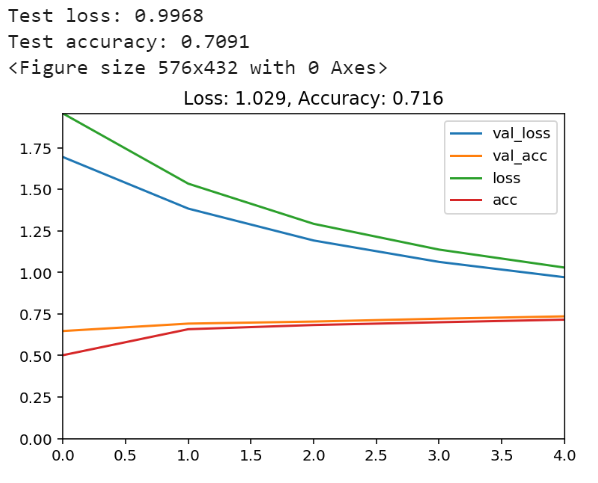
Ankle boot

For the classification of these categories, we’ve experimented with different networks (multi-layer perceptron). Initially with a single hidden layer and with deeper networks with extra hidden layers.

**Models with one hidden layer:**

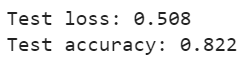
**Model 1:**

For training our dataset, we have first started with a very basic model with batch size 128 and epoch as 5 and a default learning rate of 0.001. Our basic model contains of only one dense layer with 32 units, and activation function as sigmoid. The optimizer we are using is SGD, and loss function is categorical\_crossentropy. Metrics used by us for calculating the performance of the model is accuracy. The training accuracy is 71% only thus the model has not overfitted the model, however the test accuracy is also not good enough hence we need to change the parameters of our model.



**Model 2:**

In this model we have kept the batch size as 128 and increased the epoch as 50 and a default learning rate of 0.001. Our basic model contains of only one dense layer with 32 units, and activation function as sigmoid. The optimizer we are using is SGD, and loss function is categorical\_crossentropy. Metrics used by us for calculating the performance of the model is accuracy. The test accuracy has increased with increase in the number of epochs. The training data is getting fitted well with increase in the number of epochs which resulted in increase of the training accuracy by 10%.



**Model 3:**

For this model we tried changing the batch size to 54000 with other parameters same as before. This degrades the performance and we get our test accuracy as 0.12 which is really low. With increase in the batch size our training accuracy is also decreasing. The reason for this change is, the updating of weights and gradient is very low in case of large batch sizes. Where as in case of mini-batch, the gradient along with weights will get updated quickly with decrease in training time. It has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize. The lack of generalization ability is due to the fact that large-batch methods tend to converge to *sharp minimizers* of the training function.

The batch sizes of 500 and 1000 are almost giving similar results as batch size of 256 with not much significant difference in accuracy. The only difference is the test loss which increases with 256 batch size in comparison to 500 batch size.

**Model 4:**

In this model, we have changed the learning rate parameter for SGD optimizer, and kept the batch size as 54000 and epoch as 50. Our test accuracy increases with the increase in the learning rate to 0.637. The reason might be because, we need optimum learning rate for our model accuracy to improve. Too large learning rate might lead to a long training process that could get stuck. With increase in the learning rate our training accuracy is increasing.

**Model 5:**

In this model we have changed our optimizer to “adam” and used batch size 256 with epoch as 50 and a default learning rate of 0.001. Our basic model contains of only one dense layer with 32 units, and activation function as sigmoid. The optimizer we are using is SGD, and loss function is categorical\_crossentropy. Metrics used by us for calculating the performance of the model is accuracy. It was observed that the training time with adam optimizer is more compared to the SGD.

Accuracy for the test set -> 0.87

Loss for the test set -> 0.359

**Model 6:**

For this model we have increased the learning rate for adam optimizer from 0.001 to 0.01 with same batch size and epochs as model 4.

We have the test loss as 0.561

And Test accuracy as 0.809

We can see there is decrease in our test accuracy compared to the model 4 which has learning rate of 0.001. This could be because, of the increase in learning rate which diverges from the local minimum when compared to the learning rate of 0.001.

With the learning rate of 0.005 and same parameters as model 4 and optimizer as “adam” we get almost the same test accuracy which is as below-

Accuracy for the test set -> 0.873

Loss for the test set -> 0.413

**Model 7:**

Further for this model, I have changed the optimizer to “nadam” with batch size of 500 and epochs as 50. We are getting the below result-



We found that the accuracy is increasing a bit compared to model 5 and also the test loss is decreasing by 7%.

With the decrease in batch size from 500 to 64, we can observe that the accuracy is decreasing and test loss is increasing slightly. And we found that the training accuracy has increased slightly with the batch size of 64 which might be the problem of over fitting.



**Model 8:**

In this model we have increased the learning rate of “nadam” optimizer from default value of 0.002 to 0.01 keeping the batch size as 500 and epochs as 50.

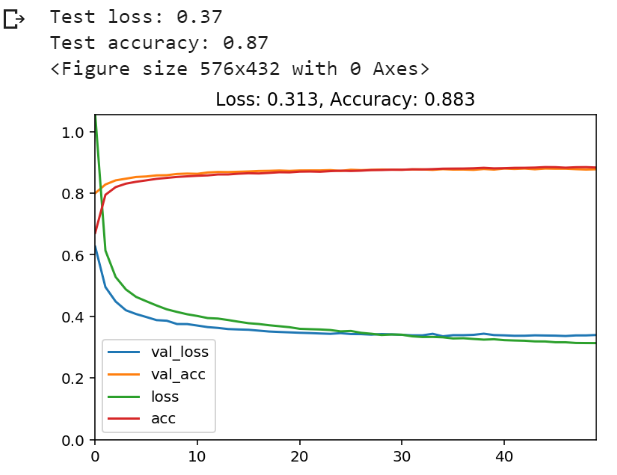


We found that the test accuracy is decreasing and the test loss is increasing with increase in the learning rate.

Next, we have used the drop-out model for fitting our dataset in order to check the test accuracy and see if the model is overfitting previously. The drop out models refers to dropping out units (both hidden and visible) in neural network.

**Model 9:**

Here we have used one dense layer with 32 units and sigmoid activation function. We are using dropout with parameter as 0.2. Batch size is 256 with learning rate of 0.002 and optimizer is nadam and epochs are 50. The accuracy and loss are not improving but it did prevent over fitting as earlier we had training accuracy of 0.91 whereas now it has become 0.883.



Next, we are using deeper model, by increasing the dense layers-

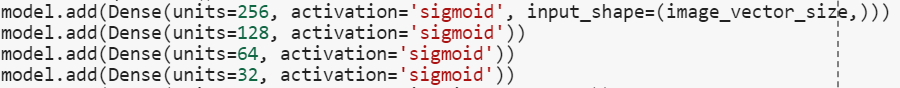
**Model 10:**

In this model, we are adding extra hidden layer with 64 neurons and checking the test accuracy of our model.



This has increased the test accuracy of the model with decrease in the test loss. Hence, we can say that adding extra layer is helping in better classification however, if we keep increasing the hidden layers the runtime will increase which will result in increased computational cost and model complexity.

Increasing more hidden layers as below–



The test loss increasing and test accuracy is decreasing very minutely but when checking the training accuracy, we found that adding more layers increases the training accuracy, this maybe because of overfitting.

**Model 11:**

In this model we have added two dense layers one with units 256 and another with 1024 units and we are using adam optimizer with batch size 256 and 50 epochs. This resulted in highly complex model with 1,068,810 parameters to be trained. The test accuracy increased in this case compared to all the previous models but it took lot of time to train the model. Thus, increasing computational cost.



We tried adding convolutional layers to check if the performance of the model is improving.

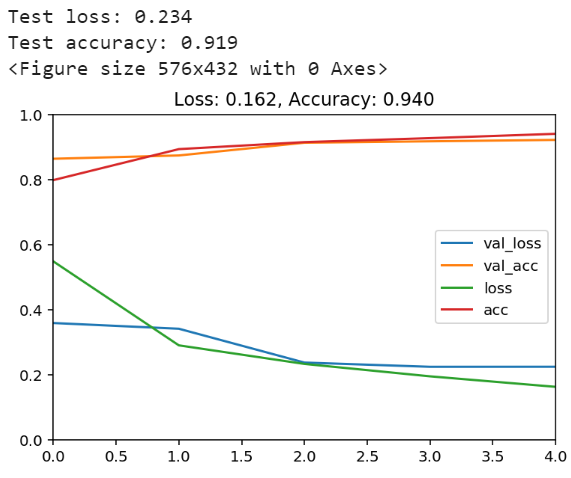
**Model 12:**

When using adagrad as optimizer and epoch as 25, with batch size as 100. Here we have used parameter as same for padding which means will append with 0 instead of dropping. Relu activation function has been used for the convolution layers. In this case, we are getting a good model with test loss as 0.444 and test accuracy as 0.916 which is is the maximum test accuracy so far however, we also see our model is overfitted giving training accuracy as 0.991 which is quite high. The plot below shows jittery lines, which as it is closely following the dataset and might not always perform well with unseen dataset. As the batch size is too small, it is expected to overfit the data and the same is happening in the model.



**Model 13:**

For this model, we have used nadam optimizer, epochs as 5 and batch size as 256. The reason of keeping epochs low, is the model takes lot of time with convolution layers. Low epochs are giving considerable good test accuracy with low test loss and we get the below result. This model is performing even better than the model 12. In this we are decreasing the epochs and increasing batch size slightly to reduce the overfitting which gives us slightly better result of 0.919 accuracy with 4% decrease in training accuracy compared to model 12.



**Findings from the above models-**

* Increasing batch size leads to poor test accuracy due to underfitting.
* For higher batch size, test accuracy increases with increase in the learning rate as this decreases underfitting and our model tries to gather few information quickly from the entire dataset.
* When we increase the learning rates it is not performing well as in most cases the default learning rates is good for the given optimisers.
* Adding convolution layers are increasing the performance, though the models get overfitted we are getting good accuracy.
* With increase in the drop outs with convolution layers our training accuracy is decreasing without effecting our test accuracy much. This way we are decreasing the overfitting.
* When increasing the value for drop out, we are actually underfitting the data thus performance is decreasing.
* Increase in the number of neurons in fully connected network increased the performance but the time taken to train the model increased.
* Thus, convolution layers are better option with some drop out layer for this dataset as it increases the performance without overfitting the dataset.
* Nadam optimizer is performing best without any convolution layer out of adam, adagrad and SGD.
* On using drop out layers with the convolution layer we see the training accuracy has reduced, thus reducing the overfitting in the model.

**Appendix:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Optimizer | Parameters | Test Loss | Test Accuracy | Training Accuracy |
| Model1 | SGD | dense layer- 32 neuron | 0.9968 | 0.709 | 0.716 |
|  |  | batch size-128 |  |  |  |
|  |  | epoch-5 |  |  |  |
|  |  | learning rate- 0.01 |  |  |  |
| Model2 | SGD | dense layer- 32 neuron | 1.0043 | 0.711 | 0.833 |
|  |  | batch size-128 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.01 |  |  |  |
| Model3 | SGD | dense layer- 32 neuron | 2.25 | 0.12 | 0.118 |
|  |  | batch size-54000 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.01 |  |  |  |
| Model4 | SGD | dense layer- 32 neuron | 1.65 | 0.637 | 0.644 |
|  |  | batch size-54000 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.1 |  |  |  |
| Model5 | adam | dense layer- 32 neuron | 0.359 | 0.87 | 0.908 |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.001 |  |  |  |
| Model6 | adam | dense layer- 32 neuron | 0.561 | 0.809 | 0.828 |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.1 |  |  |  |
| Model7 | adam | dense layer- 32 neuron | 0.756 | 0.71 | 0.726 |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.2 |  |  |  |
| Model8 | adam | dense layer- 32 neuron | 0.413 | 0.873 | 0.934 |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.005 |  |  |  |
| Model9 | nadam | dense layer- 32 neuron | 0.356 | 0.876 | 0.916 |
|  |  | batch size-500 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.002 |  |  |  |
| Model10 | nadam | dense layer- 32 neuron | 0.42 | 0.872 | 0.938 |
|  |  | batch size-500 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.01 |  |  |  |
| Model11 | nadam | dense layer- 32 neuron | 0.354 | 0.882 | 0.935 |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.002 |  |  |  |
|  |  | dropout-0.2 |  |  |  |
| Model12 | nadam | dense layer- 32 neuron | 0.383 | 0.864 | 0.855 |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.002 |  |  |  |
|  |  | dropout-0.4 |  |  |  |
| Model13 | nadam | dense layer- 32 neuron | 0.378 | 0.883 | 0.954 |
|  |  | dense layer- 64 neuron |  |  |  |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.002 |  |  |  |
| Model14 | adam | dense layer- 1024 neuron | 0.423 | 0.892 | 0.974 |
|  |  | dense layer- 256 neuron | |  |  |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.001 |  |  |  |
| Model15 | adagrad | dense layer- 1024 neuron | 0.345 | 0.881 | 0.933 |
|  |  | dense layer- 256 neuron | |  |  |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-50 |  |  |  |
|  |  | learning rate- 0.01 |  |  |  |
| Model16 | adagrad | Conv layer - 32, (3,3) | 0.323 | 0.916 | 0.987 |
|  |  | Conv layer - 32, (3,3) |  |  |  |
|  |  | max pooling - (2,2) |  |  |  |
|  |  | Conv layer - 64, (3,3) |  |  |  |
|  |  | Conv layer - 64, (3,3) |  |  |  |
|  |  | max pooling - (2,2) |  |  |  |
|  |  | dense-layer - 512 |  |  |  |
|  |  | batch size-100 |  |  |  |
|  |  | epoch-25 |  |  |  |
|  |  | learning rate- 0.01 |  |  |  |
| Model17 | nadam | Conv layer - 32, (3,3) | 0.234 | 0.919 | 0.94 |
|  |  | Conv layer - 32, (3,3) |  |  |  |
|  |  | max pooling - (2,2) |  |  |  |
|  |  | Conv layer - 64, (3,3) |  |  |  |
|  |  | Conv layer - 64, (3,3) |  |  |  |
|  |  | max pooling - (2,2) |  |  |  |
|  |  | dense-layer - 512 |  |  |  |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-5 |  |  |  |
|  |  | learning rate- 0.002 |  |  |  |
| Model18 | nadam | Conv layer - 32, (3,3) | 0.267 | 0.9 | 0.916 |
|  |  | Conv layer - 32, (3,3) |  |  |  |
|  |  | Drop Out- (0.2) |  |  |  |
|  |  | max pooling - (2,2) |  |  |  |
|  |  | Conv layer - 64, (3,3) |  |  |  |
|  |  | Conv layer - 64, (3,3) |  |  |  |
|  |  | max pooling - (2,2) |  |  |  |
|  |  | dense-layer - 512 |  |  |  |
|  |  | Drop out – (0.2) |  |  |  |
|  |  | batch size-256 |  |  |  |
|  |  | epoch-5 |  |  |  |
|  |  | learning rate- 0.002 |  |  |  |