Report

Image classification using Fashion mnist

Name: Arun kumar S

S.R.No: 15656

Department : UG

All of my solutions are using tensorflow 2.0.

I have used test data for validation in this project.

I have compiled both the networks using ‘adam’ optimizer and Sparser categorical crossentropy as the loss function. I used ‘adam’ as it is the standard optimizer used generally in Machine learning . Though we don’t have too many classes in this project , Sparse Categorical Crossentropy is still helpful and it also improves the speed of the algorithm .

Solution using a Deep multilayer network:

First, I have normalized the data by dividing all the pixel values by 255. This is done just to keep the values of pixels in a common scale without distorting the differences in the ranges of values, i.e, without losing any information. Normalization also improves the speed of the algorithm.

In my model, since I'm designing the usual deep feedforward network, I need to have my input in the form of a column vector consisting of the input values. So, first I have a layer that flattens out the 28 * 28 pixel values in the form of a column vector. I thought any other representation (ex: binary representation) of the input will not help as the relative values between the pixels decide the type of the image. My output needs to be one of the 10 classes of clothing . So, the final output layer will be a Dense layer with 10 neurons. I chose 'softmax' as the activation function for the final layer since it is the most common recommended activation function for multiclass classification. I had to decide the number of Dense hidden layers to keep in my model. First, I tried with 1 hidden layer with 128, 32 or 256 neurons. I found that with one layer the training and the test accuracies are around 0.86 with training accuracy being

greater than test accuracy each time. I have tried these with various number of epochs and with various values of dropout and considered the best results.

I have a Dropout layer in all of my architectures which I used to tune the trade-off between overfitting and underfitting in each case. I have done regularisation using Dropout in this project.

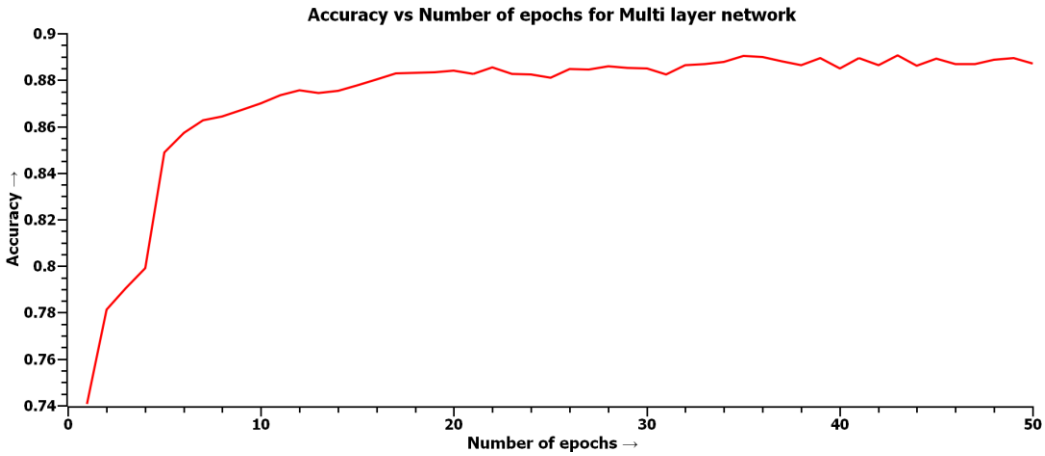
when I tried with 3 hidden layers, the accuracies in both training and test images is relatively less compared to the accuracies that I obtained with one hidden layer. I have tried 3 hidden layers with various epochs and dropout values and with various combinations of number of neurons. Below I have attached a table containing some of the best results that I got using different number of layers with different combinations of number of neurons. The number of neurons that I have used in any layer in any architecture are of the form 2^N. Keeping the number of neurons in this fashion improves the time taken by the algorithm and also improves the results. When I tried with 2 hidden layers, I got on an average, better results compared to both other cases of one hidden layer and 3 hidden layers. So, it gave a hint that I can find my best architecture and hyperparameters with 2 layers. So, I tried various combinations of neurons for 2 layers ((128,32), (256,64), (128,64), ...). I finetuned the number of epochs and the dropout value in each case to get the best result in each architecture. Comparatively, the combination of 128 and 64 neurons was giving the best results. So, I finetuned this using dropout and number of epochs to get the best training and the test accuracies. I got training accuracy of 0.8871 and test accuracy of 0.8763. The number of epochs is 50 and dropout is 0.2. Trying with different combinations sequentially we have arrived the best accuracies for the Image classification problem using multilayer network.

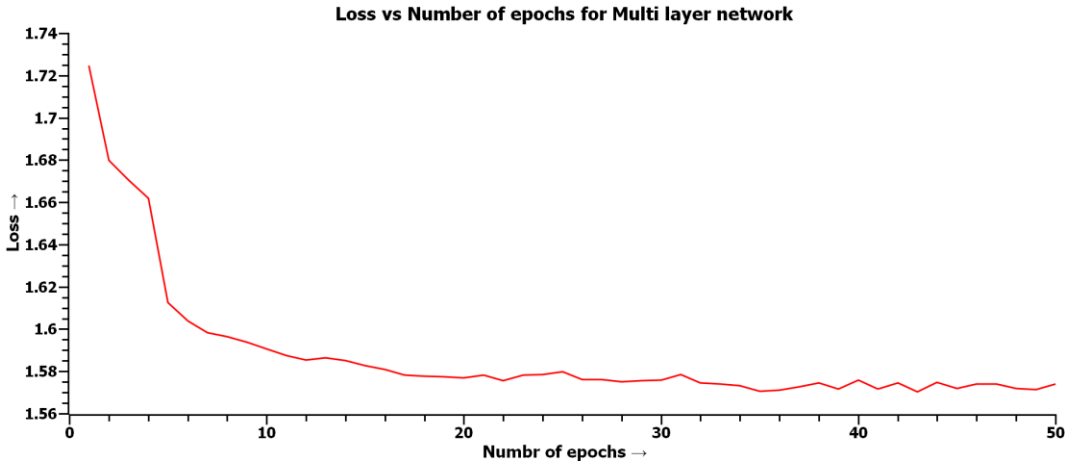
Below is the table of some of the results of different architectures that I tried using multilayer network .

Number of hidden layers	Number of Neurons (in order)	Number of epochs	Training accuracy	Validation accuracy
1	128 , Drop(0.2)	10	0.8748	0.8612
1	128, Drop(0.3)	20	0.8517	0.8651
1	256,Drop(0.17)	60	0.8304	0.817
1	32,Drop(0.15)	40	0.8242	0.8117
2	256, Drop(0.2) , 64	20	0.8612	0.8527
2	128, Drop(0.2) , 32	50	0.8802	0.8645
2	128, Drop(0.2) , 32	75	0.8803	0.8807
2	128, Drop(0.2) , 32	100	0.8805	0.8601
2	128, Drop(0.35) , 32	100	0.8629	0.86
2	128,64,Drop(0.2)	50	0.8871	0.8763
3	256,Drop(0.1),64,32	20	0.851	0.8409
3	128,Drop(0.2),64,32	15	0.1124	0.3037
3	128,64,Drop(0.2),32	60	0.862	0.8464
3	128,64,Drop(0.21),32	60	0.886	0.8704

Here Drop(0.x) corresponds to a Dropout layer with 0.x dropout rate.

Below are the graphs for accuracy vs number of epochs and loss vs number of epochs for the model which has been saved(i.e that gave best results).





Both loss and accuracy nearly saturate after some epochs . I have chosen number of epochs such that validation accuracy is also better along with training accuracy . Also , I have found that training accuracy doesn't improve much after a certain number of epochs which means that the model has reached its best with the given model capacity . So, we need a different model that processes the data differently to achieve better accuracies . CNN is one such model .

Solution using Convolutional Neural network:

I have again normalized the pixel values to keep the values in the same scale without losing any information and improving the speed of the algorithm.

Output is again one of the 10 classes of clothing . So, output layer will be a dense layer with 10 neurons with 'softmax' activation function for the same reason as described above . I had to decide the number of convolution layers . Usually a convolution layer consists of one layer with filters and another layer with Pooling . After passing through convolution layers we need to flatten out the final values and pass it through a Dense layer and finally the output layer to give out the probabilities of the 10 classes of clothing . I have used a Dense layer with 128 neurons (since it gave good results) with standard 'relu' activation function for this purpose . With 2 convolution layers (with different

combinations of number of filters) , training accuracy and test accuracy were around 0.91 and 0.89 . I have used (3,3) filters in most of my CNN architectures as they gave good accuracies . I have used (2,2) pooling always for the same reason . With 3 convolutional layers , the accuracies went down . Training and test accuracies were around 0.86 and 0.85 .Obviously this will not be the desired architecture . With one convolution layer the training and the test accuracies were 0.92 and 0.9 on an average . So, this is even better than the results that I found with the case of 2 convolutional layers . I tried various number of filters with one convolutional layer . I found that the one with 64 (3,3) filters and (2,2) pooling gave the best accuracies in both training(0.928) and test data (0.9078) . So, my saved model has training accuracy of 0.928 and test accuracy of 0.9078 . Below is table with some of the results that I got in the different architectures of CNN that I tried .

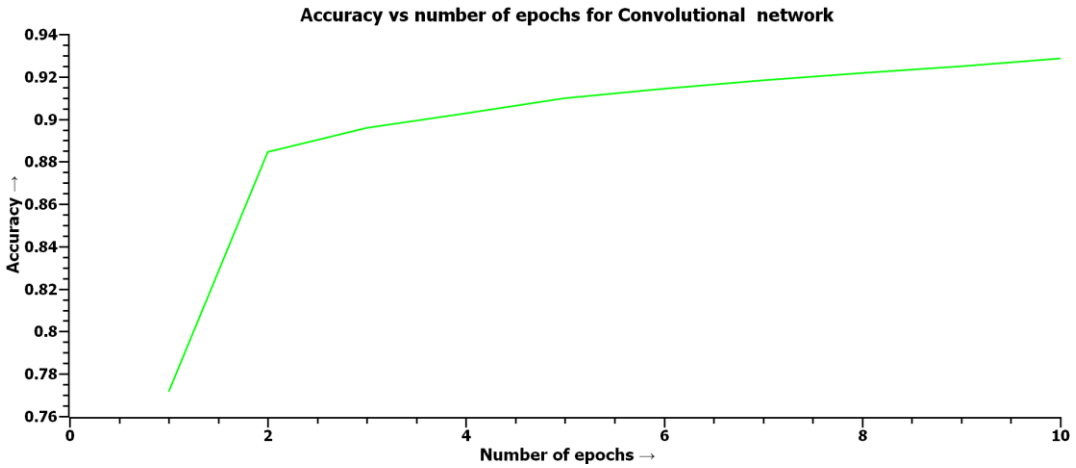
Number of Conv layers	Filters, Pooling (order)	epochs	training accuracy	validation accuracy
1	C(32,(3,3)),Drop(0.1),P(2,2)	10	0.927	0.9052
1	C(64,(3,3)),Drop(0.12),P(2,2)	10	0.928	0.9078
1	C(128,(3,3),Drop(0.2),P(2,2)	10	0.9233	0.9008
1	C(64,(2,2)),Drop(0.15),P(2,2)	10	0.926	0.9004
1	C(32,(2,2)),Drop(0.1),P(2,2)	10	0.9248	0.9003
2	C(64,(3,3)),P(2,2),C(64,(3,3)),P(2,2)	10	0.9098	0.8894
2	C(32,(3,3)),P(2,2),Drop(0.2),C(32,(3,3)),P(2,2)	10	0.8954	0.8931
2	C(64,(3,3)),P(2,2),Drop(0.2),C(32,(3,3)),P(2,2)	10	0.9254	0.8932
2	C(64,(3,3)),P(2,2),Drop(0.2),C(32,(3,3)),P(2,2)	7	0.9193	0.9004
2	C(64,(3,3)),P(2,2),Drop(0.1),C(32,(3,3)),P(2,2)	10	0.9269	0.9061
2	C(64,(3,3)),P(2,2),C(32,(3,3)),P(2,2)	10	0.9289	0.898
2	C(64,(3,3)),P(2,2),Drop(0.15),C(32,(3,3)),P(2,2)	10	0.9239	0.9002
2	C(64,(3,3)),P(2,2),Drop(0.25),C(32,(3,3)),P(2,2)	10	0.9193	0.8998
3	C(64,(3,3)),P(2,2),Drop(0.2),C(64,(3,3)),P(2,2),C(32,(3,3)),P(2,2)	10	0.8692	0.8536
3	C(64,(3,3)),P(2,2),Drop(0.2),C(32,(3,3)),P(2,2),C(32,(3,3)),P(2,2)	10	0.8582	0.8506
3	C(128,(3,3)),P(2,2),Drop(0.2),C(64,(3,3)),P(2,2),C(32,(3,3)),P(2,2)	10	0.8672	0.8423

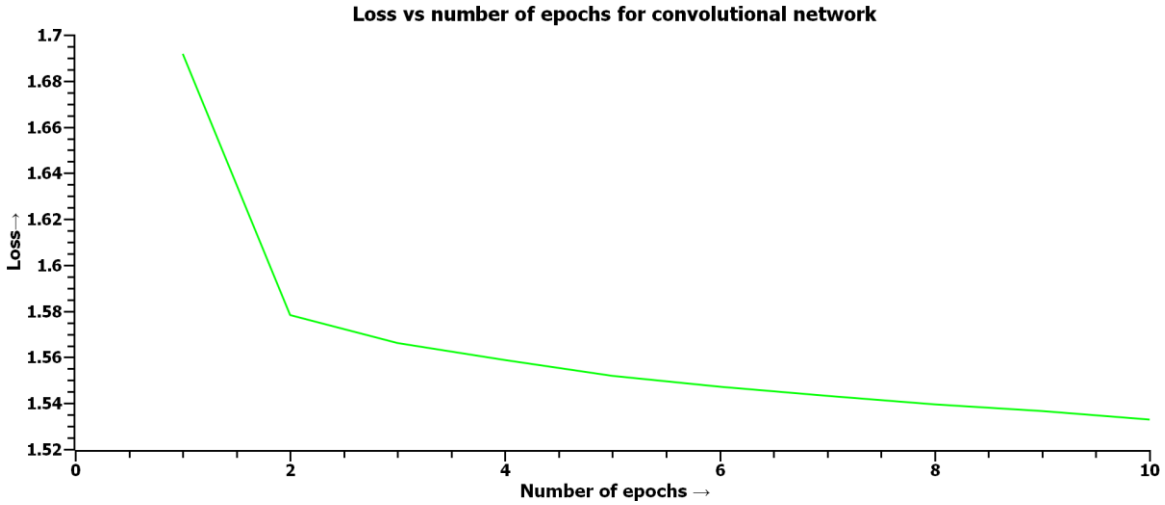
Here, C(x,(y,y)) – Convolution layer with x (y,y) filters .

P(x,x) is x by x pooling layer .

Drop(0.x) is a dropout layer with 0.x dropout rate .

Below , I have attached the graphs of accuracy vs number of epochs and loss vs number of epochs .





It may seem that accuracy and loss are still improving with number of epochs but after this there is overfitting and So I have stopped after 10 epochs . CNN model processes the images in an entirely different way and leads to best results with just 10 epochs (still time for each epoch is larger with CNN model) .

Multilayer network vs Convolutional neural network :

Convolutional neural network gives better results when compared with the usual deep multilayer network . This may be attributed to the different representations in the 2 different architectures . CNN uses filters to extract the features of the image for predicting the output whereas Multilayer network just uses the pixel values of the image to find out the class of the image . Filtering out unwanted parts of the image and looking at the defining features of the image should yield a better result when compared to the case where we

need to find the underlying distribution by processing the 28 * 28 numerical values . For the deep multilayer network to improve I think we need to build a very large neural network and the amount of data should also be very high . It is a very difficult task to learn the classification of images using only pixel values with only 60000 images for training for the multilayer network. CNN processes the data in an entirely different way using filters . I think that the main advantage of CNN is that it filters out unwanted pixel values and extracts only the useful information . But multilayer network needs to process all the pixel values . CNN is having accuracy more than Multilayer network by 2-3 % nearly . But when the task is much more complex (involving images of course), then CNN will show significant better results when compared with the multilayer network .