Deep Learning – Autumn 2019

Assignment 1 – Building Neural Network Architectures

PART B – Computer Vision Neural Network Building

Part 1

Transfer Learning

**About Dataset:**

There are 60,000 examples in the training dataset and 10,000 in the test dataset and that images are indeed square with total 784 pixels.

**Dataset Reshaping:**

we have to convert into 28X28 pixels through reshape. We also updated the pixels from 28 to 32 pixel through padding.

**Modelling:**

Technically, CNN models to train and test each input image will pass through a series of layers with kernals, pooling and fully connected layers. We also applied softmax activation function to classify an object with probability between 0 and 1. While, Compile defines the loss function, the optimizer and the metrics.

**resNet:**

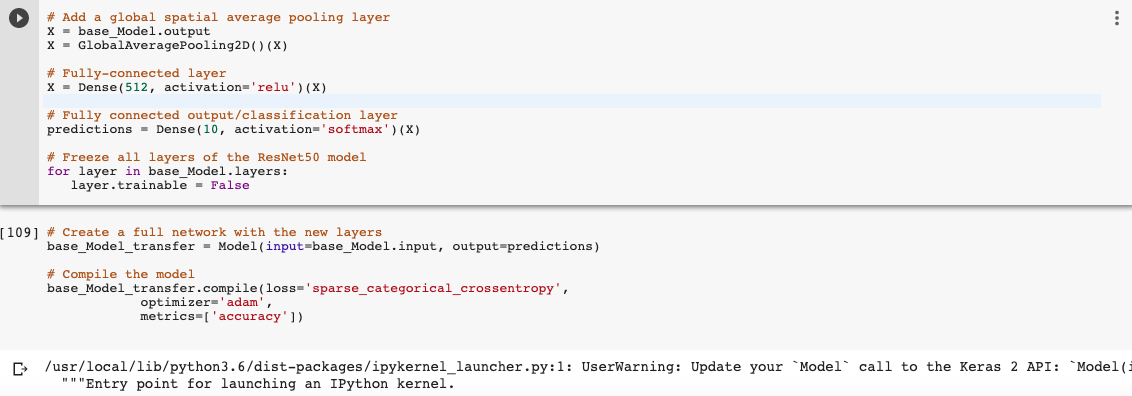
The resnet network then does the following:

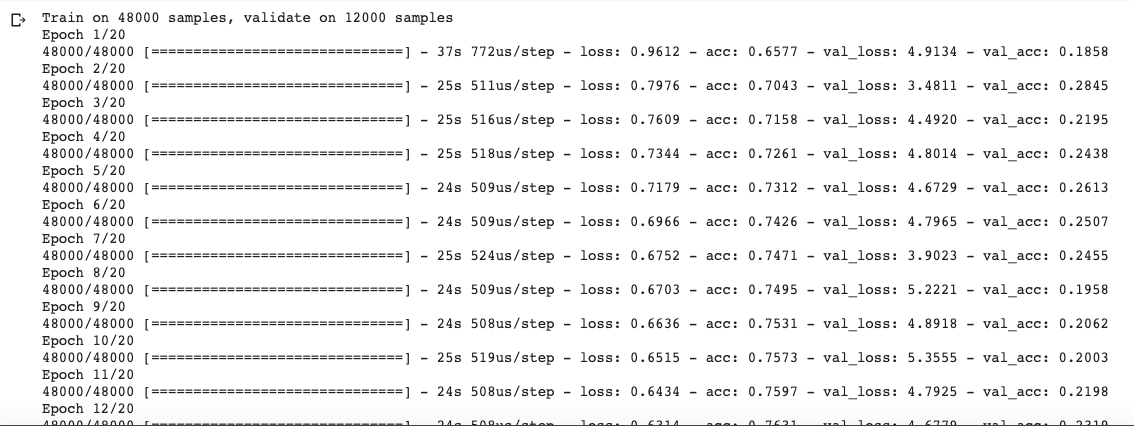
zero pad input image conv2d (valid), batchnormalised, relu activation zeropadding again maxpooling Then starts repeating conv blocks with identitiy blocks in stages.

**Prediction using resnet50:**

At first glance, resnet50 without any modification to predict from fashion mnist dataset. We load ResNet50 model which is already pre-trained weights as per the keras benchmark. We add a global spatial average pooling layers and fully connected layers. The dense layer type is use in for our output layers. We will have about 10 nodes in our output, one for each possible outcome.

**Transfer learning:**



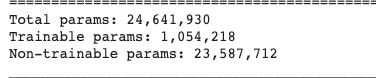


all base\_model layers where frozen -- mean that the new model will take the final output of the base, do all initial feature detection and the more complex ones (this is a problem) before applying my dense layers, the final result was very bad with val\_acc == .1

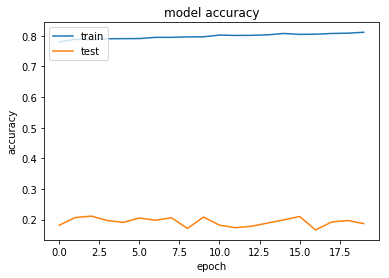
The first attempt is not working properly so decided to redo the experiment without freezing the layes. i did try to freeze all but last two, but that did not help. and i run this one on a smaller data set to see results quickly.

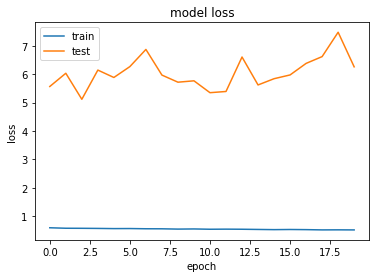
At first time we got around test accuracy of about 6.44.

After the regularization with previous model, we improve our model. this case was training 24 million parameters.



Even after added another FC layer at the end, trained on the full data set to deal with variance issue, and trained. training accuracy plateaued quickly, after 20 epochs, while val accuracy remained very low, around .18 validation loss did not drop below 5.6 compared to training loss at .59 so far, it seems that the resnet weights are heavily trained to recognise the data set features, our data set seems to be very different for transfer learning to be able to adapt to it.





**Create a New CNN with 7 Layers:**

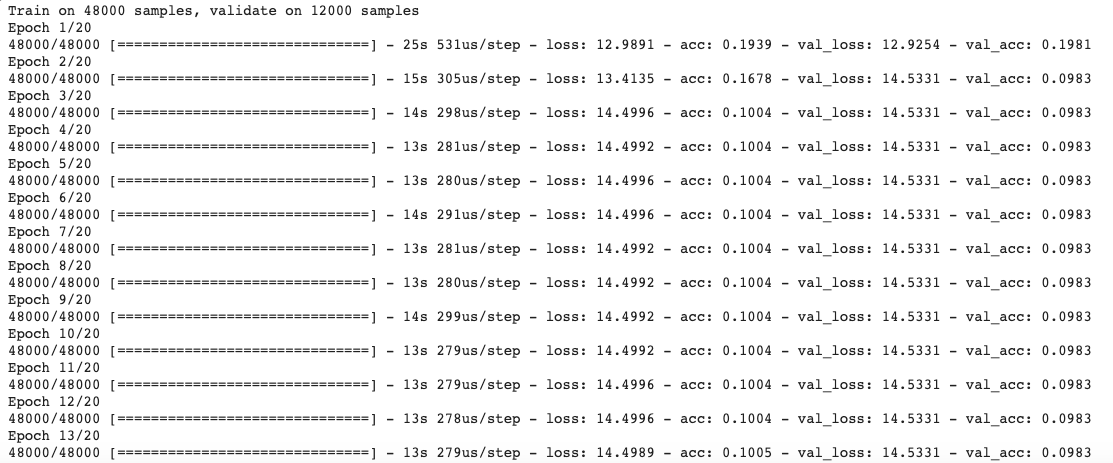
Here, We created custom CNN model with 7 layers. Here we use Dropout as to prevent the overfittng in the model. Random connections are dropped to ensure that no single node in the network is responsible for activating when presented with a given pattern. We used softmax function to find the probability between 0 and 1 and ‘sparse categorical crossentropy’ as a loss function.



Our model has two sets of layers (CONV2D –> RELU -> Maxpool). These layers are also include dropout. We used ‘Adam’ optimizer which is controls the learning rates. The adam optimizer adjusts the learning rate throughout training. The learning rate determines the how fast the optimal weight for the model are calculated. A smaller learning rate may lead to more accurate weight, but the time it takes to compute the weights will be longer.

‘Categorical crossentropy’ is the most commonly used for the classification. A lower score indicates that the model is performing better. To check the result, we will use the ‘accuracy’ metric to see the accuracy score on the validation set as we train the model.

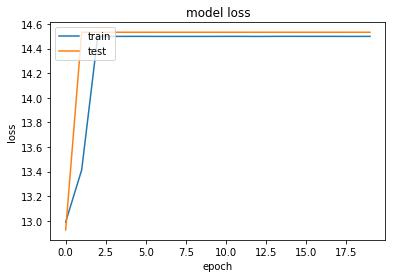
We train our model using ‘fit()’ functions. For validation data, we used the test set provided. The number of epochs is the number of times the model will run through the dataset. The more epoch we run, the more the model will improve to certain points. We used here 20 epochs. Batch size is used to devide the dataset into number of the size, therefore it is easier to run the model and faster. After 20 epochs, we got about 0.0983 validation accuracy.

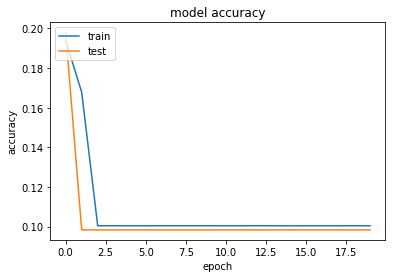


This time we are training around 300 thousand parameters.



This time after running 20 epochs, we got accuracy around 0.1004 and loss is about 14.49 while validation loss and validation accuracy are 14.53 and 0.0983 respectively.





**Enhancement:**

This time after running 12 epochs, we got accuracy around 0.1001 and loss is about 14.50 while validation loss and validation accuracy are 14.51 and 0.0995 respectively when we are using ‘SGD’ as a optimizer.



