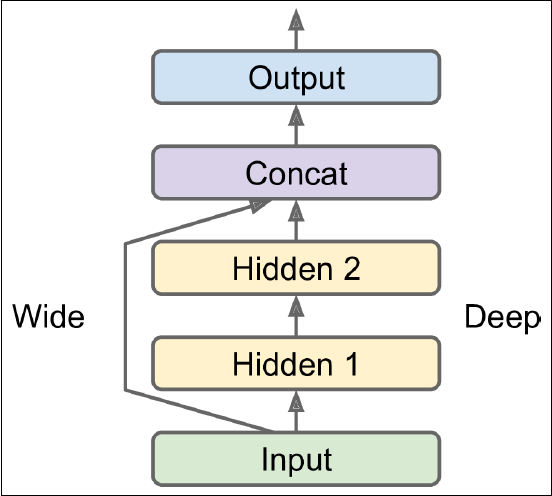
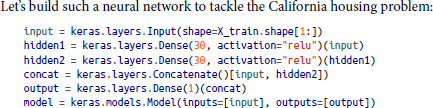
Name: Jogesh Venkata Surya Prakash Devathi

WsuId: n659a624

# Assignment #3: Deadline: 11/19/2020

**(Points: 100)**

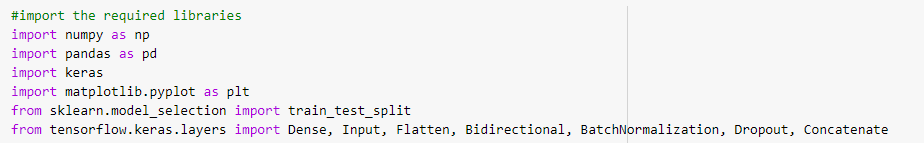
**Q1** Please use the Fashion MNIST dataset in ‘NeuN\_Part\_I code’ to implement the following network along with batch normalization and dropout layers. Please refer to page 305 of your ML book for the example code on another dataset (also given below).



# Deliverables

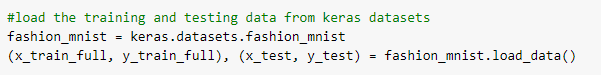
1. Well documented code
2. Your observation and analysis on the impact on network training and validation accuracy.

**Step 1**: import the required libraries



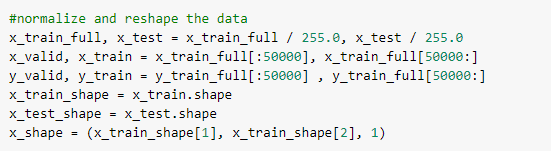
**Step 2**:

Load the dataset Fashion\_MNIST from keras and divide the test and train dataset from it.



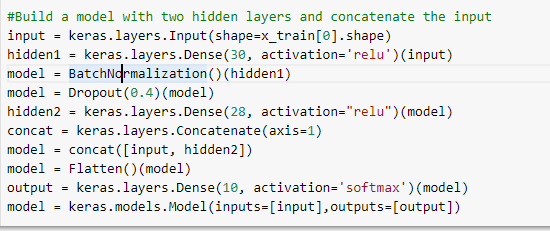
**Step 3**:

Since the image data in x\_train\_full is from 0 to 255 , we need to rescale this from 0 to 1.To do this we need to divide the x\_train\_full by 255 . It's important that the training set and the testing set be preprocessed in the same way and reshape the data.



**Step 4**:

Now we build a model. The model has 2 Convolution layer and 2 hidden layers. Dropout() is used to avoid over-fitting. BatchNormalization is used to normalize the output of the previous layers. The activations scale the input layer in normalization. Using batch normalization learning becomes efficient also it can be used as regularization to avoid overfitting of the model.



**Step 5**:

Compile the model

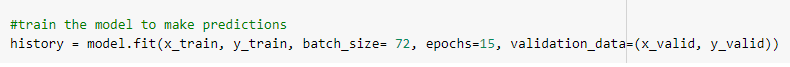
Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

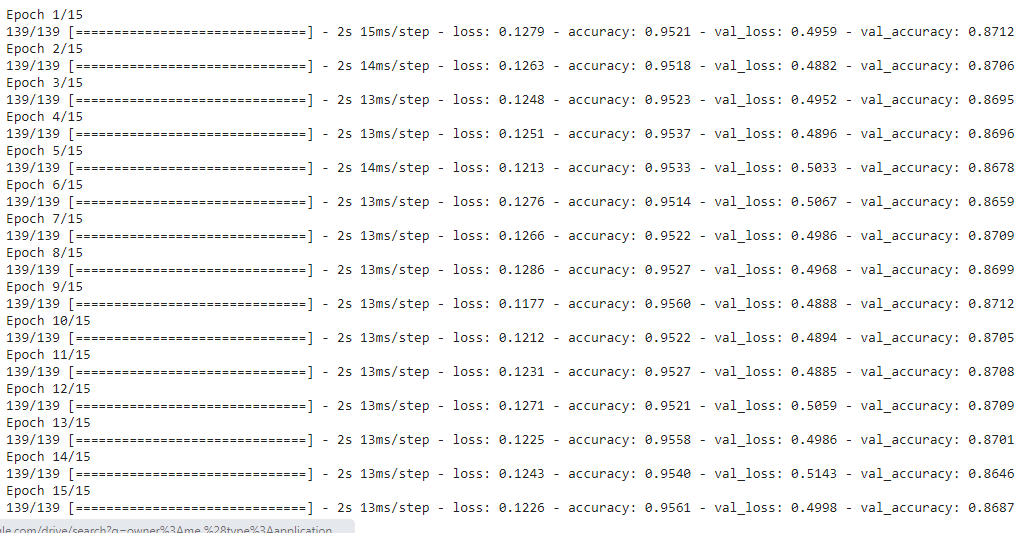
* Loss function —This measures how accurate the model is during training. You want to minimize this function to "steer" the model in the right direction.Here we will use "sparse\_categorical\_crossentropy".
* Optimizer —This is how the model is updated based on the data it sees and its loss function.
* Metrics —Used to monitor the training and testing steps. The following example uses accuracy, the fraction of the images that are correctly classified.



**Step 6**:

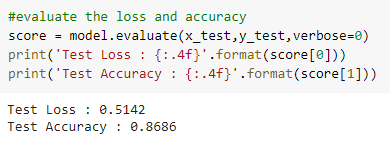
Now we fit the model with the data. You can increase the accuracy by increasing the number of epochs, Conv layer. The main advantage of this is the amount of Data is reduced.





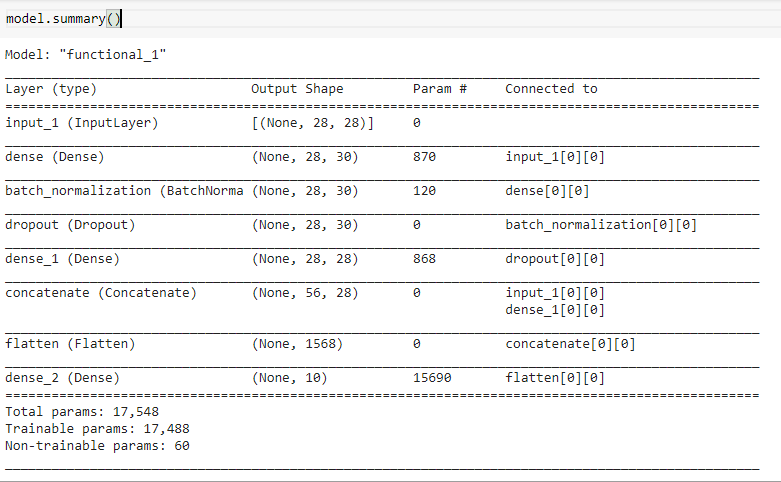
**Step 7**:

Evaluate the model and predict accuracy.



**Step 8**:

Model Summary is as follows.



**Observation and analysis on the impact on network training and validation accuracy: -**

* Using Normalization and Dropout functions reduces overfitting.
* Batch size does not affect your accuracy. This is just used to control the speed or performance based on the memory in your GPU. If you have huge memory, you can have a huge batch size so training will be faster. Even trying with different Batch Size value accuracy didn’t change.
* Having a very large epoch size may improve your accuracy. Epoch sizes can increase the accuracy up to a certain limit beyond which you begin to overfit your model. Having a very low one will also result in underfitting. As a rule of thumb, when you notice the accuracy stops increasing, that is the ideal number of epochs you should have usually between 1 and 20. 100 seems too much already.

|  |  |
| --- | --- |
| **Epoch value** | **Accuracy (with test data)** |
| 5 | 84.49 |
| 10 | 85.4 |
| 15 | 86.16 |
| 20 | 86.08 |