**Implementation**

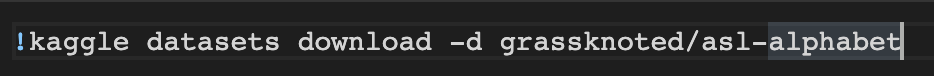
**Collecting the dataset**

The asl data set is provided by a kaggle which comprises 29 classes of hand images belonging to *x* = {a, b,c,d………………….z,space,blank}.

The image shape is inconsistent and has a mean of (70.452,71.75). Each observation in the dataset is a 3 channel image i.e. the image is of RGB value.

Collection of the data requires a kaggle api key to download the dataset.

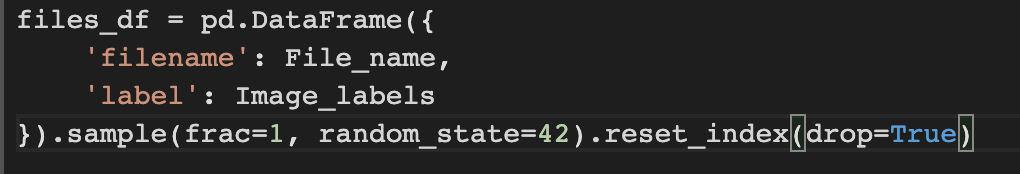
It is stored in kaggle.json file and the console downloding requires an API token.



**Preparation of dataset**

All the classes of the ASL hand gestures are in seperate folder of namly

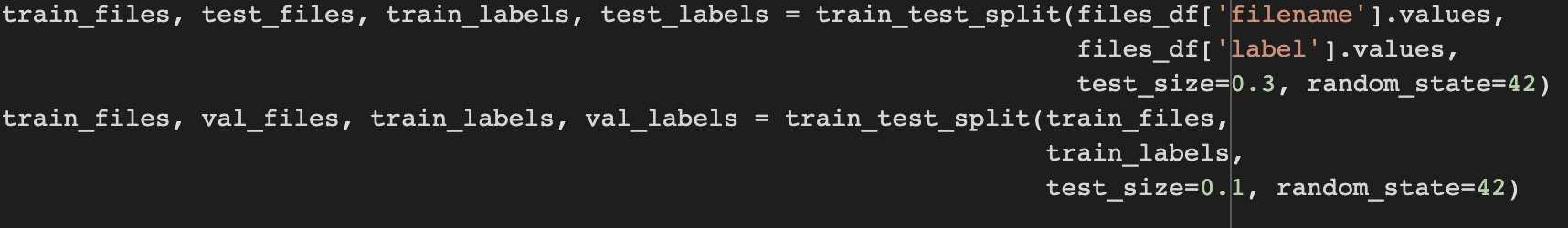
A\_hand, B\_hand…. etc these folders are needed to be extracted.

For the preparation of dataset all the files are collected in a single dataframe object (df) 

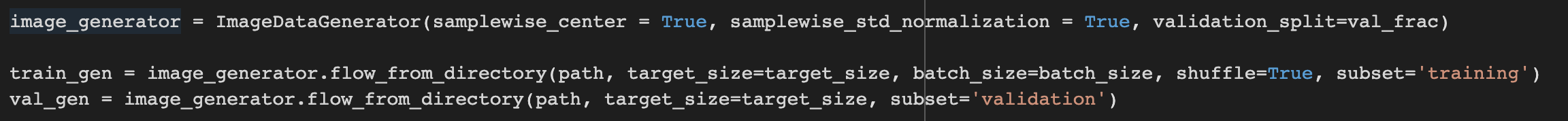
**Splitting the dataset in train, validation and test subsets**

This is achieved by using a module in python SKlearn.py and a `train\_test\_split()` function.

The dataset is divided into 70% percent training dataset and 30% test dataset. From the training dataset another 10% split is generated for the validation dataset.



Because of the large size of the dataset and for the fast and light processing of the images a generator is created for the creation of dataset.



There are 78300 images in the training dataset and 8700 images in the validation dataset.

**Preparation of Image data generator**

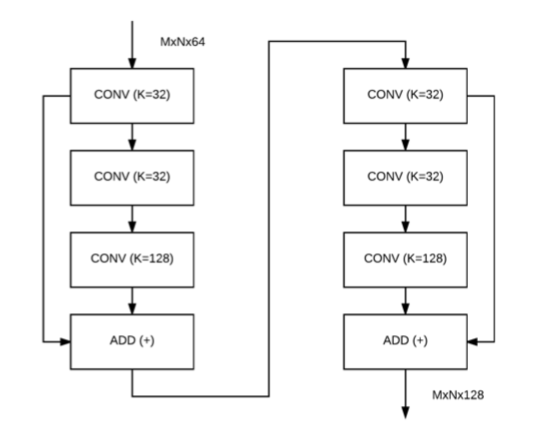
Images are resized to the shape of (1,64,64,3) , validation frequency is set to 0.1% and the batch size is set to 64.

train\_gen = image\_generator.flow\_from\_directory(path, target\_size=target\_size, batch\_size=batch\_size, shuffle=True, subset='training')

val\_gen = image\_generator.flow\_from\_directory(path, target\_size=target\_size, subset='validation')

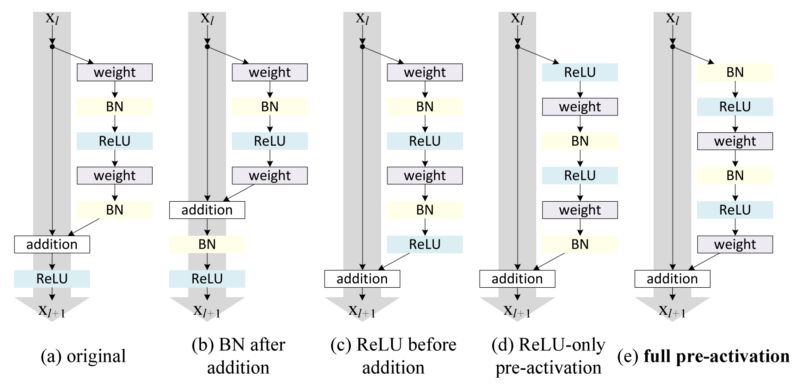
**Implementation of Model**

Implementation of ResNet50 requires to define a self contained bottle neck. That would reduce the dimensions in the first two CONV layers and increase it again in the final CONV layer. For example the figure shows 2 residual modules stacked on top of each other.



Finally, He et al. published a second paper on the residual module called *Identity Mappings in Deep Residual Networks /cite{residual } .*It provided a per-activated residual model which is a much better version of a residual block; it simplifies the flow of gradients to propagate to any previous block without hindrance.

We start with the series of (BN => RELU => CONV) \* N layers instead of starting with the convolution weights. The next residual block gets the addition operation of previous residual modules.



The overall architecture is similar to figure

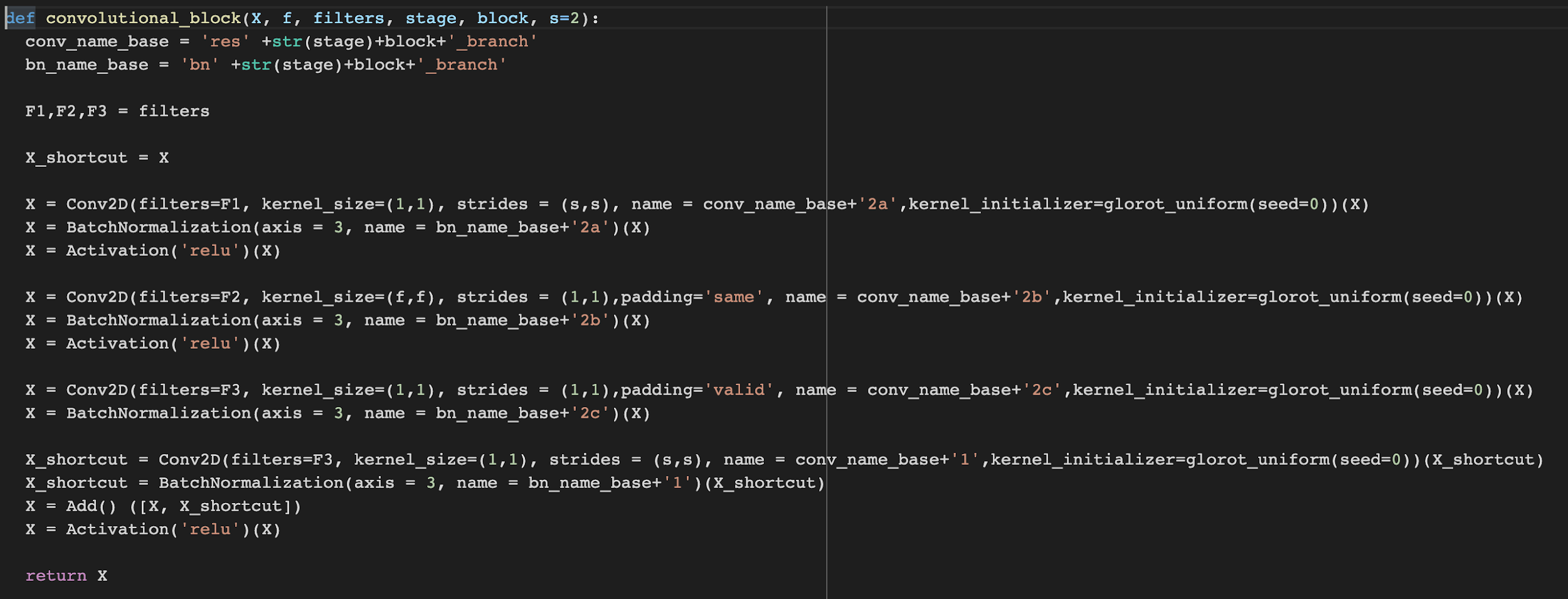
**Defining the identity block**

It comprises three identical convolutional layers with a stride size of (1,1) , kernel size of (1,1) and activation function used here is ReLU.



**Defining convolutional block**

Each Convolution block comprises four identical convolutional layers with the stride size of (1,1), kernel size of (1,1) and activation function as ReLU.

****

**Defining ResNet50**

The input layer shape is set to (1,64,64,3) and the padding is set to be zero in general until explicitly changed.

Stage one, comprises of one convolutional layer with 64 filters and and the kernel size is set to (7,7) , a batch normalization layer and the activation function is set to ReLU and finally a maxpool layer with a window size of (3,3)

Stage two, comprises a convolutional block with 64, 64 and 256 filters and it contains two additional identity blocks with 64,64,256 filters in stage ‘a’ and stage ‘b’.

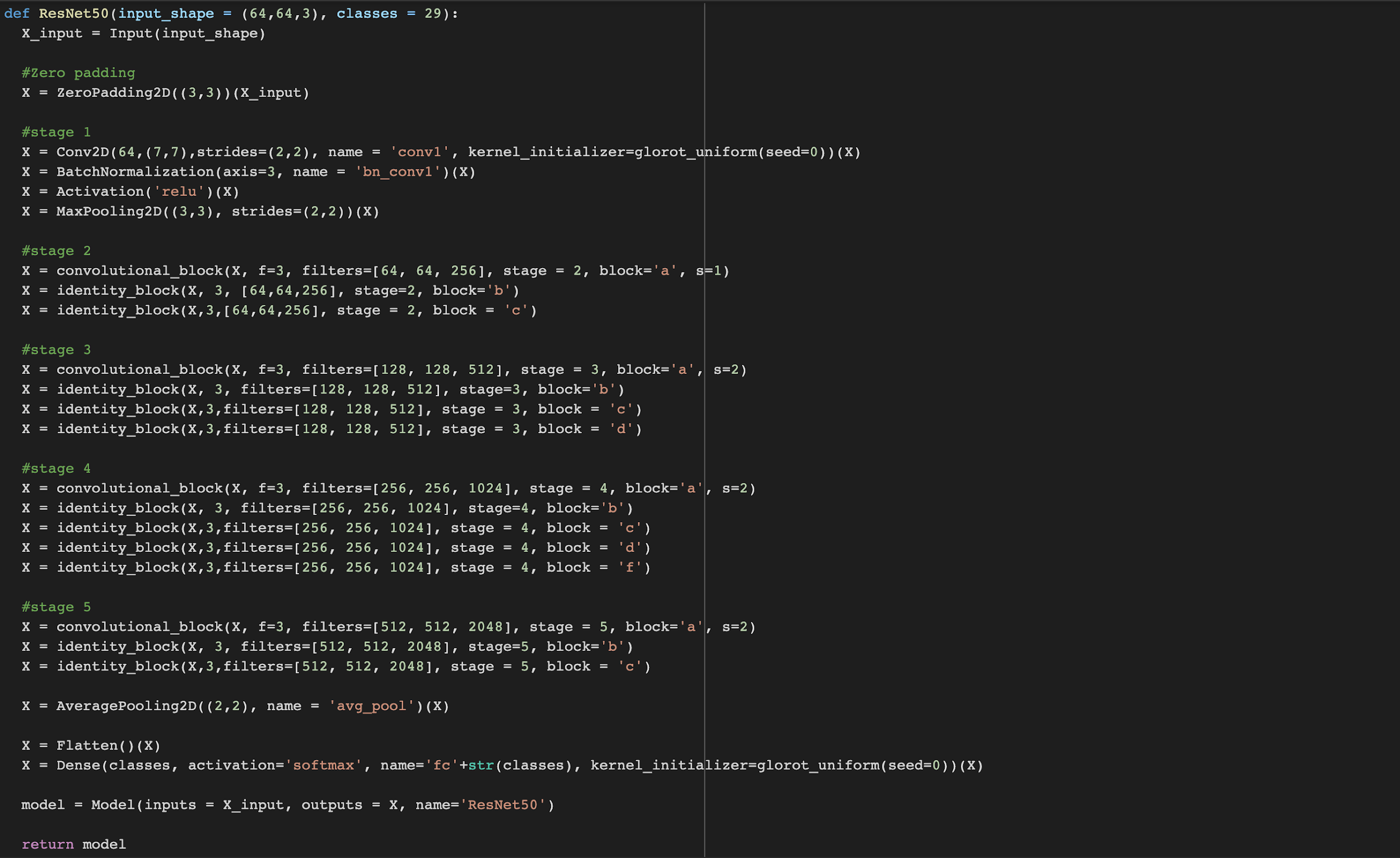
Stage three, comprises a convolutional block with 128,128 and 512 filters in three convolutional layers respectively. Followed by 3 identical identity blocks each having a filter size of 128,128 and 512 filters in each of the convolutional layers of an identity block.

Stage four, comprises a convolutional block with 256,256 and 1024 filters in each layer of convolution block respectively. Followed by four identity blocks each having 256,256 and 1024 number of filters in each of the convolutional layers respectively.

Stage Five, comprises a convolutional block with 512,512 and 2048 filters in each of the convolutional layers. Followed by two two identity blocks each having 512, 512 and 2048 filters in the convolutional layers of the identity block.

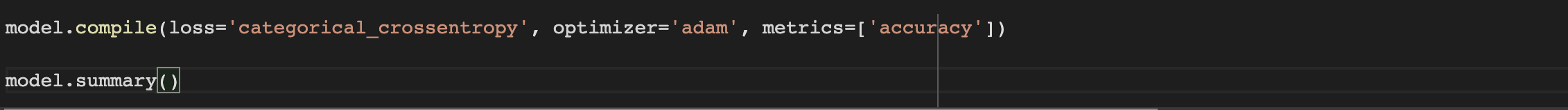
Then, the model after all five stages has an Average pooling layer followed by a flatten layer.

Finally, the model has a dense layer consisting of 29 neurons in the output layer signifying all the 29 classes of classification domain in this context.



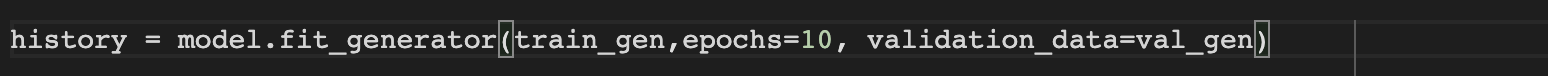
**Compilation of Model**

Model is compiled with loss set to categorical cross entropy , the optimizer is Adam to reach the saturation and the matrix to check is the accuracy.

****

**Training of model on the dataset**

Training is done using the ‘.fit’ function provided by the tensorflow library. The dataset is passed using a generator for easy computation.

****