A) Loading Dataset

First we need to find out how to deal with our data and how to get it ready to be used in the training section.

Since our data is a set of images with 10 different labels (classes). It’s a little different from the dataset we’ve used in previous tasks. The most important thing that needs to be paid attention to is to check the number of images for each of the classes. This should be done on both of the training and testing set.

B) Converting into a data frame

In this section I took some of the available information of each image and stored it as a form of a data frame. The images’ path, their class and class ids are stored here. In this way, we’ll be able to have access to the images’ path and their labels easier.

C) Resizing Images

Resizing images is a process that must be done before any model training. I converted all of the images' size into 16 * 16, for both of the training and test images. Also in this part we divided the columns of our data frames into feature and target parts. In the end, they're converted into numpy array.

D) Model

- K-fold Cross Validation

  It's a good choice to use a kind of k-fold cross validation suitable for our data. Here, we're using stratified folds which ensures that the class distribution of the original dataset is maintained throughout all of the folds. So, the balance of the classes is not changing.

- Call Backs

  Here we're using an adaptive learning rate schedulers that monitors the training loss. Using this, we're able to dynamically adjust the learning rate if the loss doesn't improve for some number of epochs, which is called the patience of training. After that, the learning rate is reduced to help our model converge faster.

- Classifier

Since our task is a multi-class image classification, we need a CNN model with multiple convolutional layers. This convolutional layers can be followed by max pooling and dropout for better results. Max pooling reduces the dimension of the feature maps and drop out prevents overfitting. Each of the convolutional layers has a filter of a specific size and before each layer we apply batch normalization.

- Training

  Now it's time to train the model on our data. We here have 5 different folds and we train our model on each of them for the number of 6 epochs while monitoring some important metircs such as training loss, training accuracy, validation loss and validation accuracy. The set batch size is 64 and data in each fold is divided into train and validation set.

E) Evaluation

In this section, we evaluate our model’s performance on the given testing set of images to figure out how well we’ve done.

The accuracy for our trained model is around 70 in this case, which is acceptable.

The End