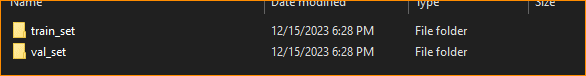
In the data set, we have multiple types of images-

Click here - <https://github.com/sadeepj/eth-80/releases/download/0.0.1/eth-80.tar.gz>

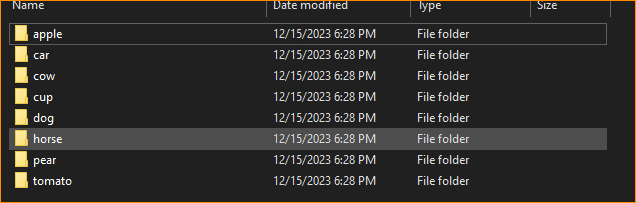
Code -

!wget 'https://github.com/sadeepj/eth-80/releases/download/0.0.1/eth-80.tar.gz'

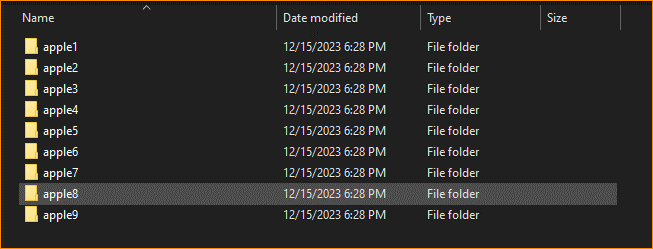
Extract that file -



Go to train set – In here have 8 classes



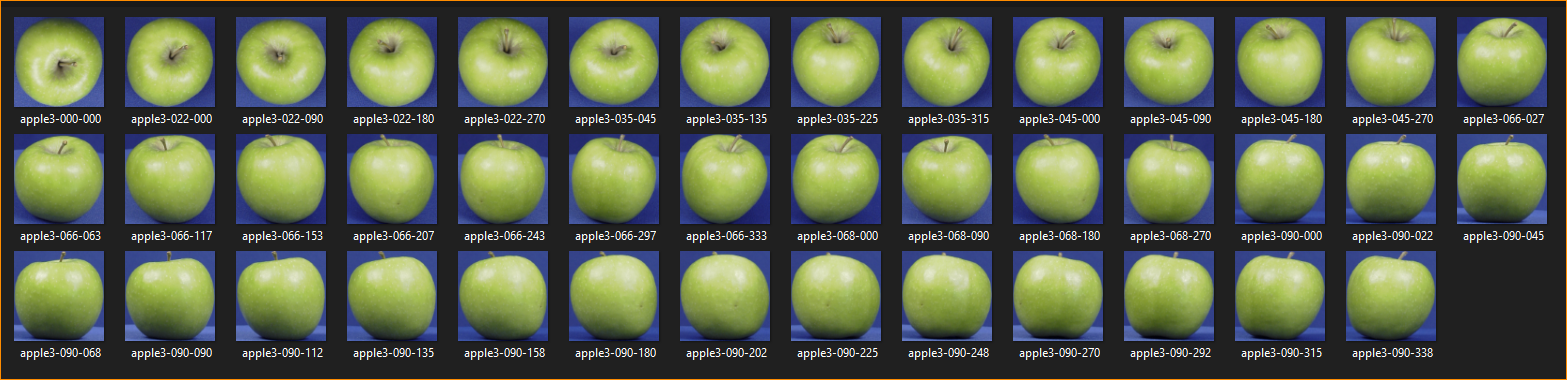
Open Apple -



Set of images – Ex – Apple

In here we are using different variations, So that mean CNN can learn accurately.







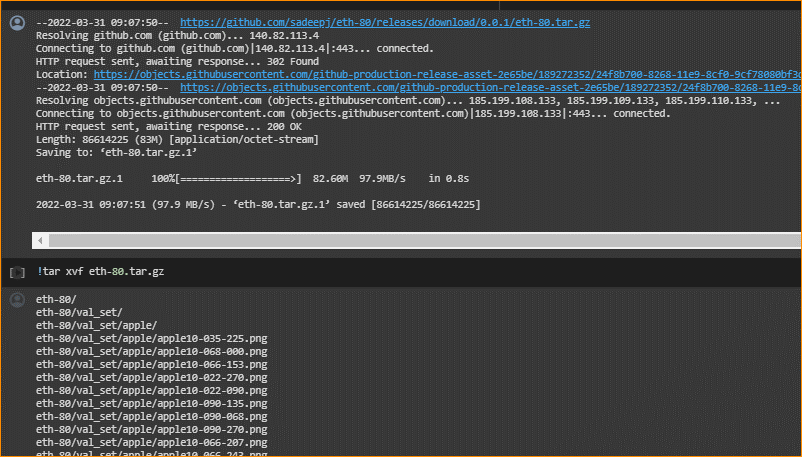
And Another file call validation set – val\_set

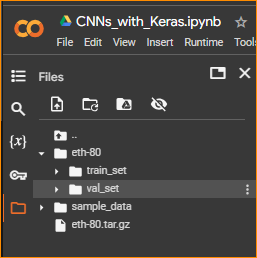
In this use the test the model – how good the model



Code-

1. Download the dataset and extract the data

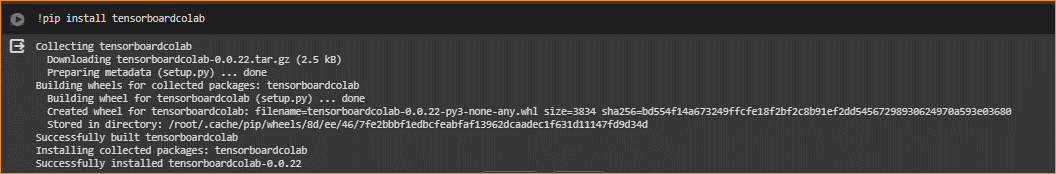


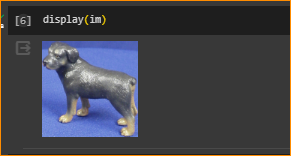


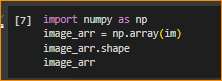
1. Verify that the data is there

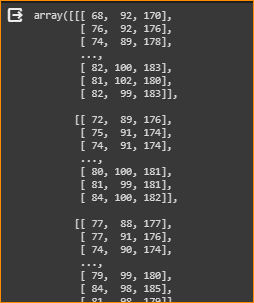




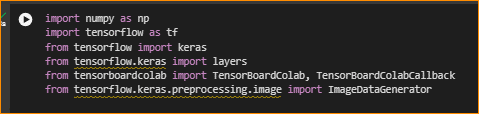






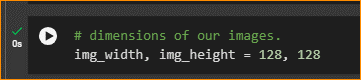


1. Module import and variable initialization



Using Libraries in TensorFlow

1. Setting the image size



1. Taking training data and validating data as a directry

Set number of samples that use to train and test

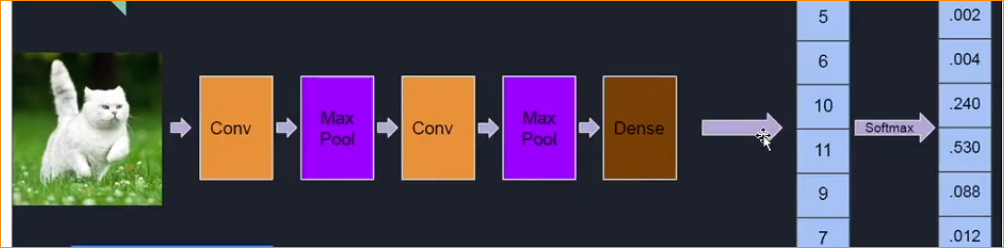
Epochs - an "epoch" refers to one complete pass through the entire training dataset. During an epoch, the model processes every training example in the dataset once and updates its weights based on the computed errors.

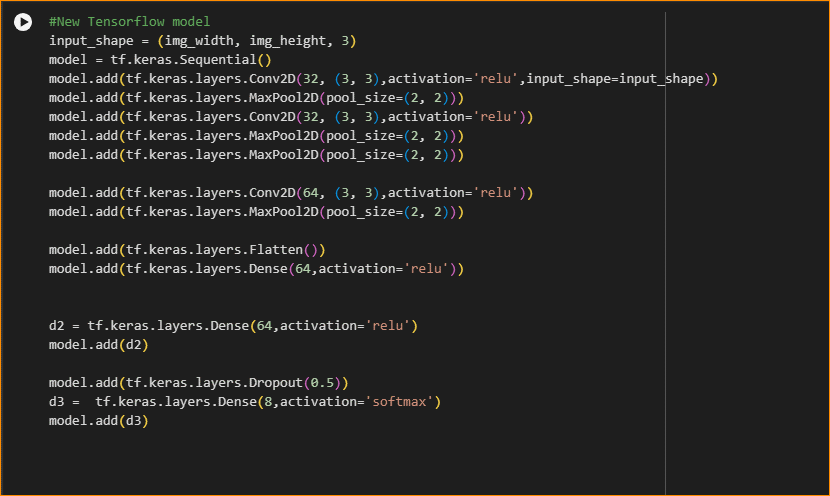
Batch\_size –

batch\_size is a hyperparameter that defines the number of training examples utilized in one iteration. In other words, it is the number of data points that the neural network processes before updating its weights. (taking random dataset)

1. Define the CNN model

Process like this –





model.add(tf.keras.layers.Conv2D(32, (3, 3),activation='relu',input\_shape=input\_shape))

* Adding convolution layer and setting the activation function.

model.add(tf.keras.layers.MaxPool2D(pool\_size=(2, 2)))

* Adding pooling layer

model.add(tf.keras.layers.Flatten())

* And flatten it

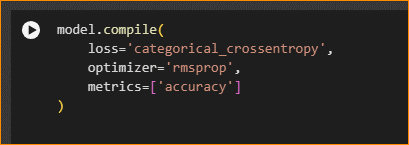
model.add(tf.keras.layers.Dense(64,activation='relu'))

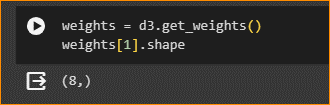
* Creating a dence(Fully connected layer)

d3 =  tf.keras.layers.Dense(8,activation='softmax')

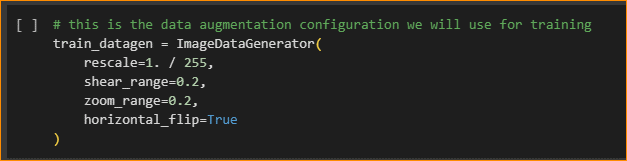
* Using softmax to creating probability solution

1. Setting the loss function



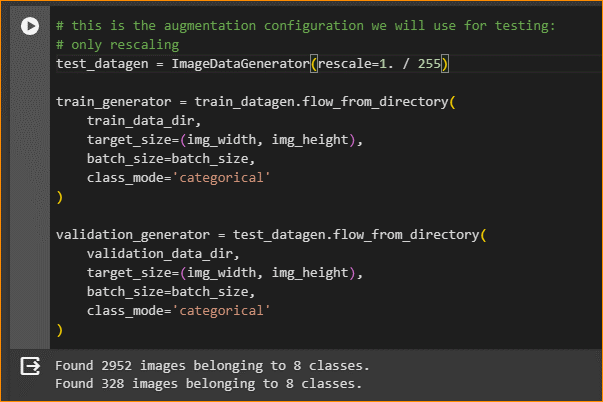


Prepare data feeders

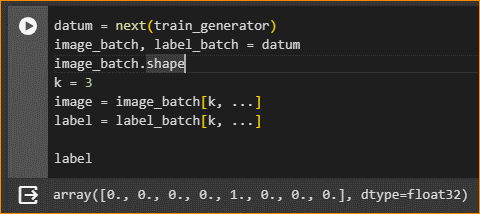


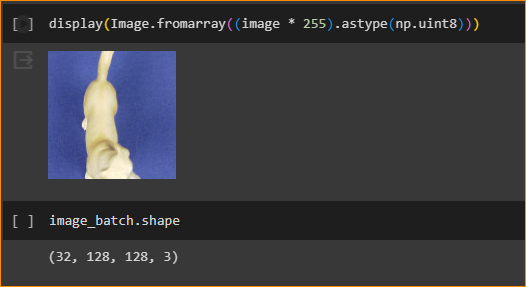
Convert pixel value range between 0 -255

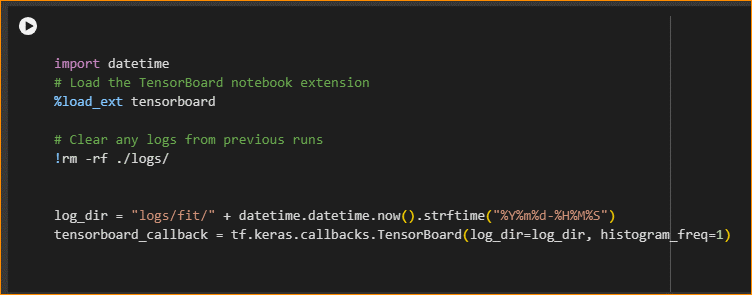
Test the generator-



1. Test the data feeders

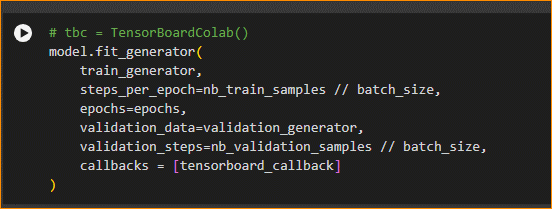


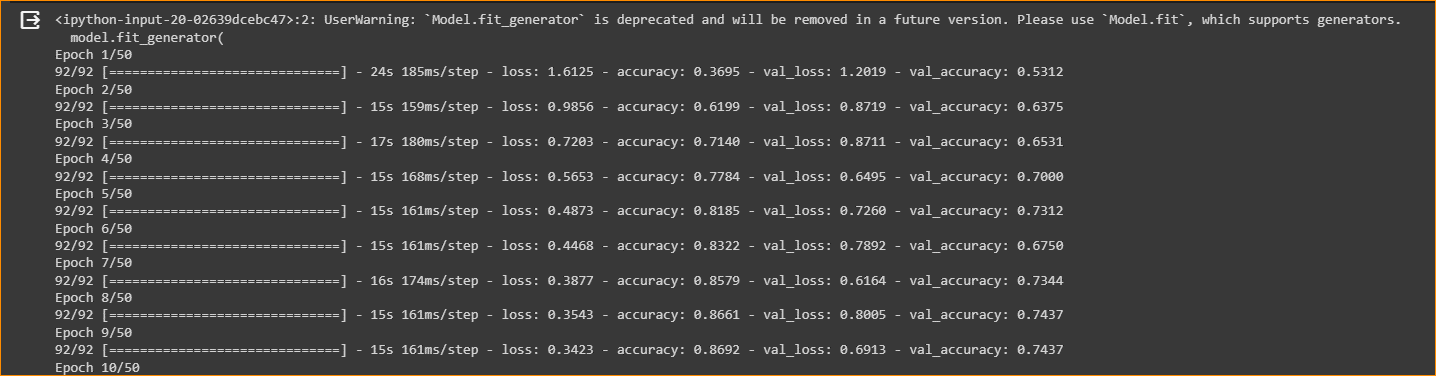


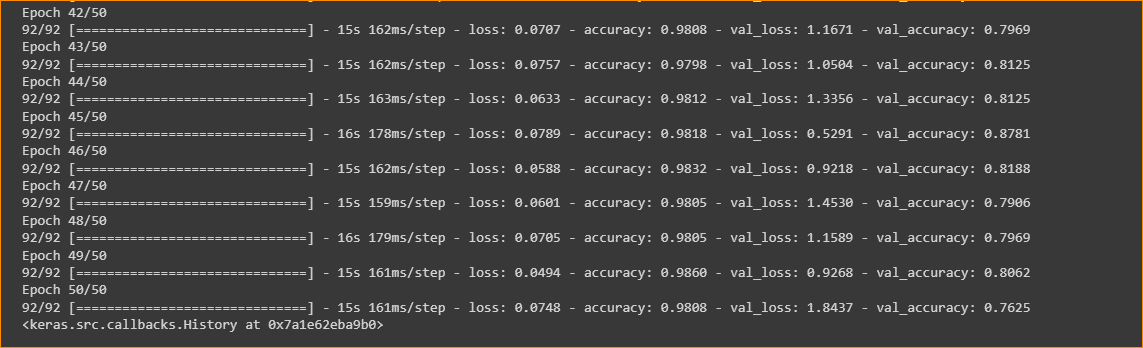


1. Fit the model

Here start the training.



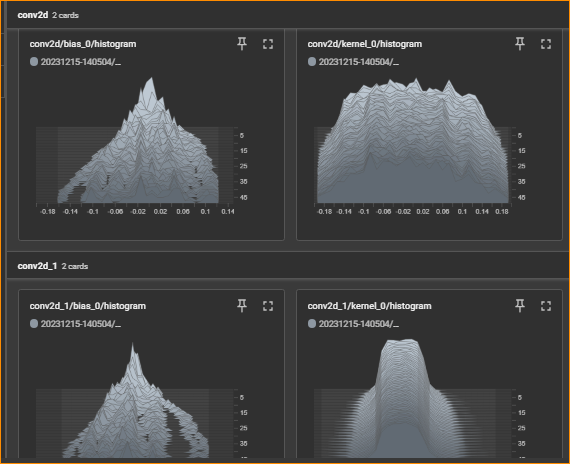


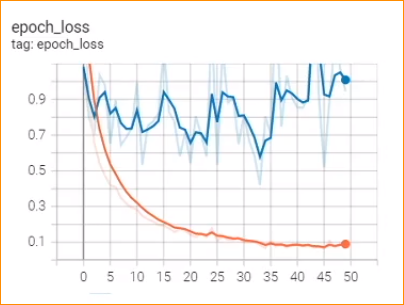


In here loss value go down and accuracy going up

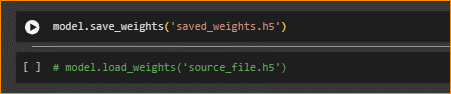
1. TensorBoard

This show accuracy with respect to the same dataset





1. Save the weights

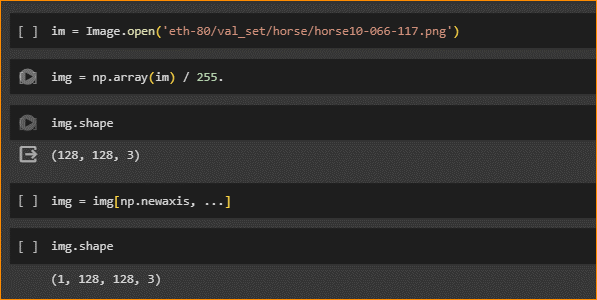


Here we save the model – h5

1. Make predictions with the trained CNN!

In here we use horse image

And do normalization / 255





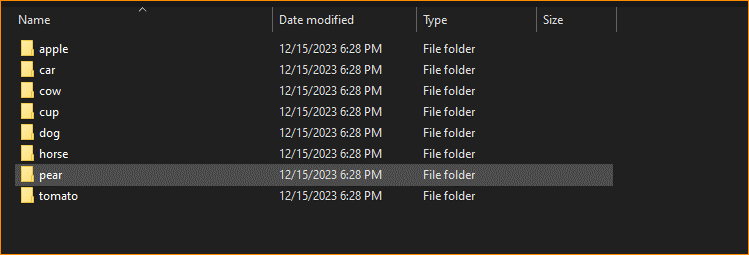
Out – mean Output of the predict values



In here 6 is Close to One(1)

Other values close to 0

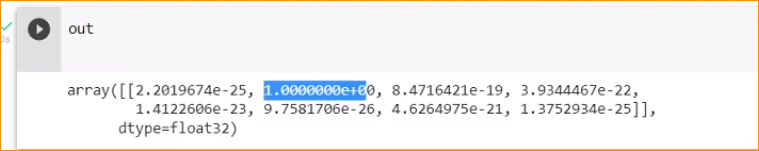
Check what is 6 Class –



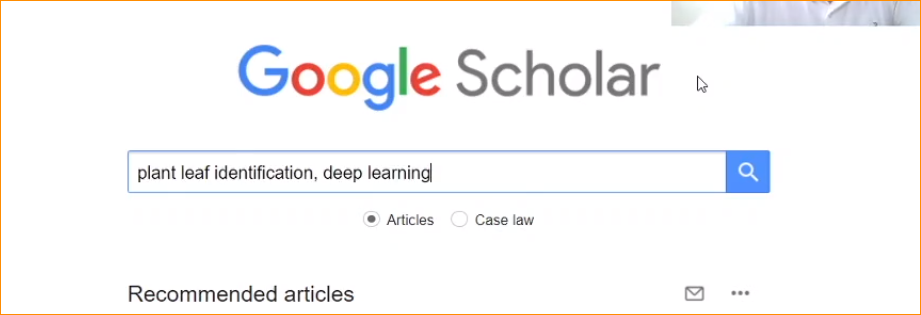
That is horse.

Now Let’s Change the image –

If you put car image –



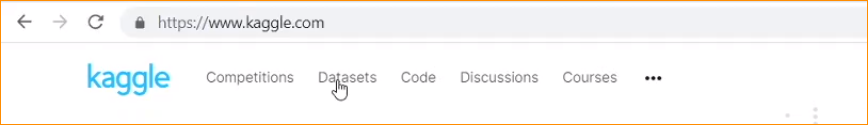
Get data set using Google Scholar-





Can see there are lot of research papers –

Using those can learn deep learning models that they are used.



Kaggle is another good resources – can download dataset