Jorge Orlando Miranda Ñahui
Github:https://github.com/jorgepdsML

CONVOLUTIONAL NEURAL NETWORKS USING PYTHON AND KERAS

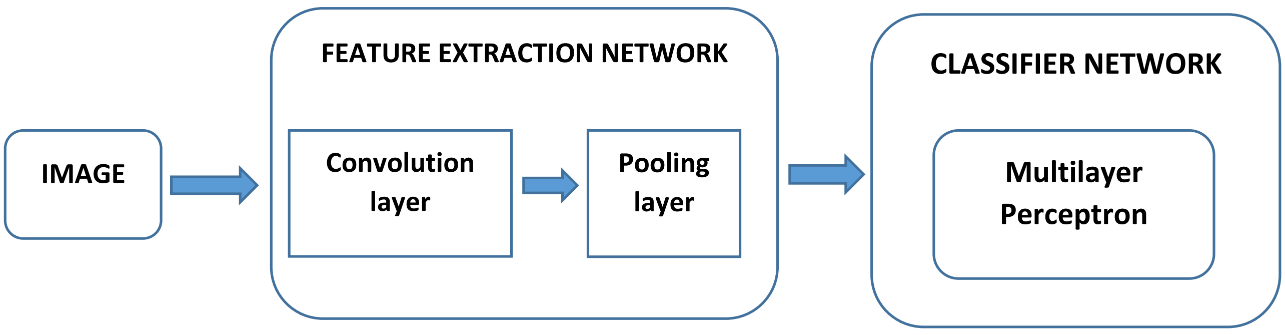
ARCHITECTURE:

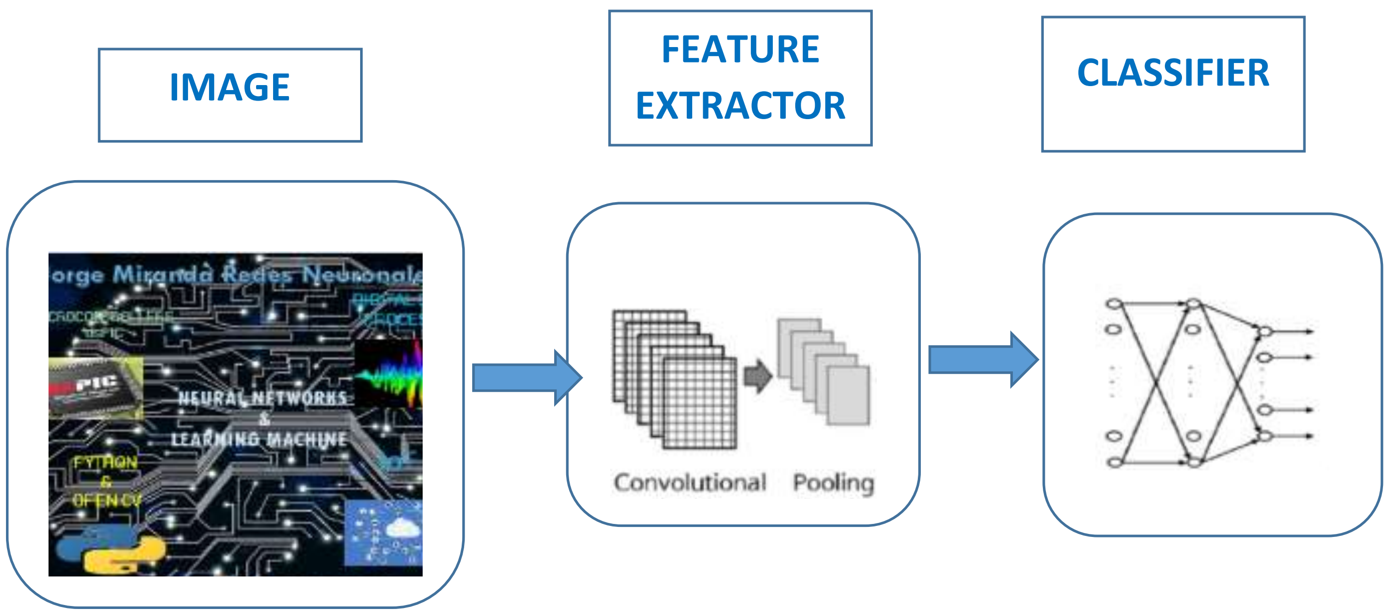
Convolutional network is a specific artificial neural network topology that is inspired by biological visual cortex and tailored for computer vision tasks by Yann LeCun in early 1990s.

According to Yann LeCun , the convolutional neural network is a specific artificial neural network topology that is inspired by biological visual cortex .it was tailored for computer vision purposes by Yann LeCun in early 1990s.

The classification of images, is based on recognizing an object based on certain features that describes the object. For example, recognizing whether the image of a picture is a man or woman is the same as classifying the image into a man or woman class.

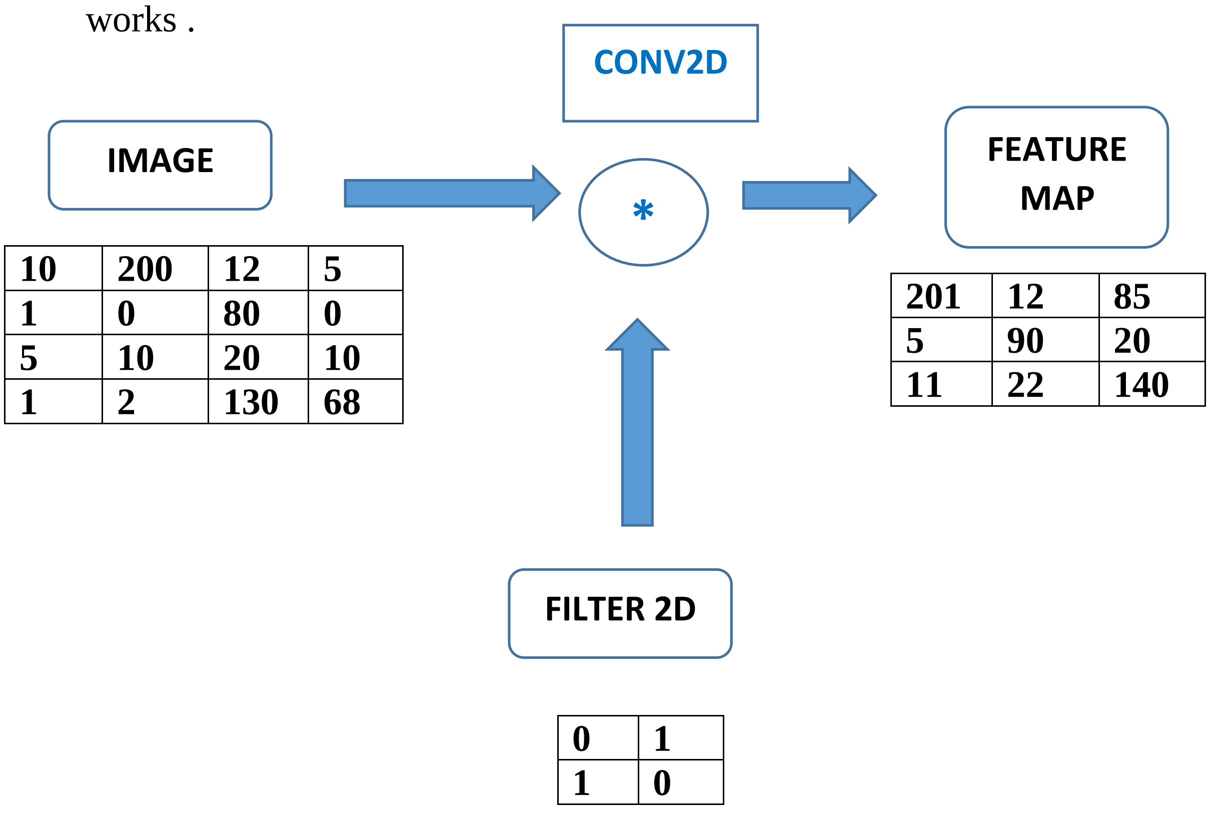
The CNN consists of feature extractor and Classifier networks, the feature extractor extracts the important features of the image based on the convolutional and pooling layer then the features extracted are classified by the Classifier, in order to make the CNN to learn the desired response for each image, its parameters will be updated at each iteration so as to minimize a specified cost function. the measure of the neural network's error is the cost function. The greater the error of the neural network, the higher the value of the cost function is.





CONVOLUTION LAYER:

The convolution layer consists of a set of filters or kernels that converts images. The purpose of the convolution layer is to come up with the feature maps that accentuates the unique feature of the original image. The feature maps are created by the convolution layer using the convolution operation to the original image. The convolution operating was originated in the field of signal processing and the understanding of its extent to the image processing is helpful to grasp the way the CNN works .

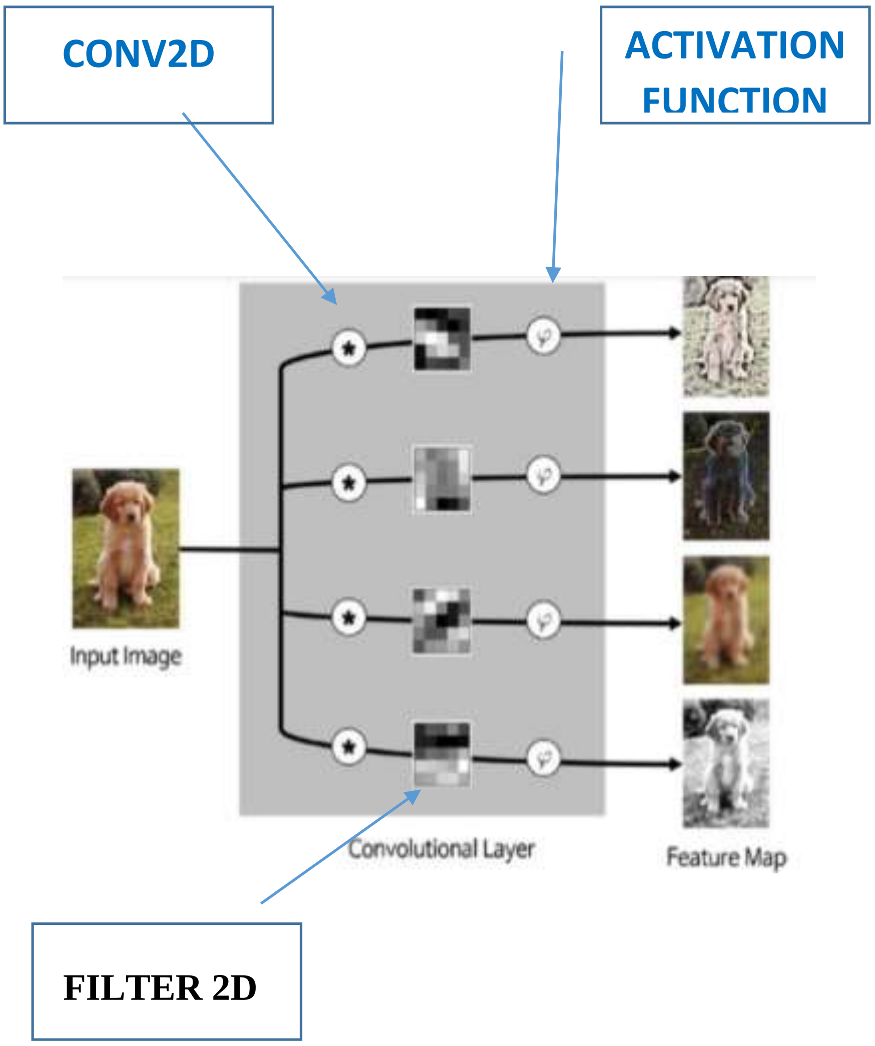


10	200	12	5
1	0	80	0
5	10	20	10
1	2	130	68

201	12	85
5	90	20
11	22	140

0	1
1	0

Usually in the convolution layer could be composed of many filters with each filer generating a feature map via the convolution operation as was explained before. All this feature maps will be input to the pooling layer what will be explained next.

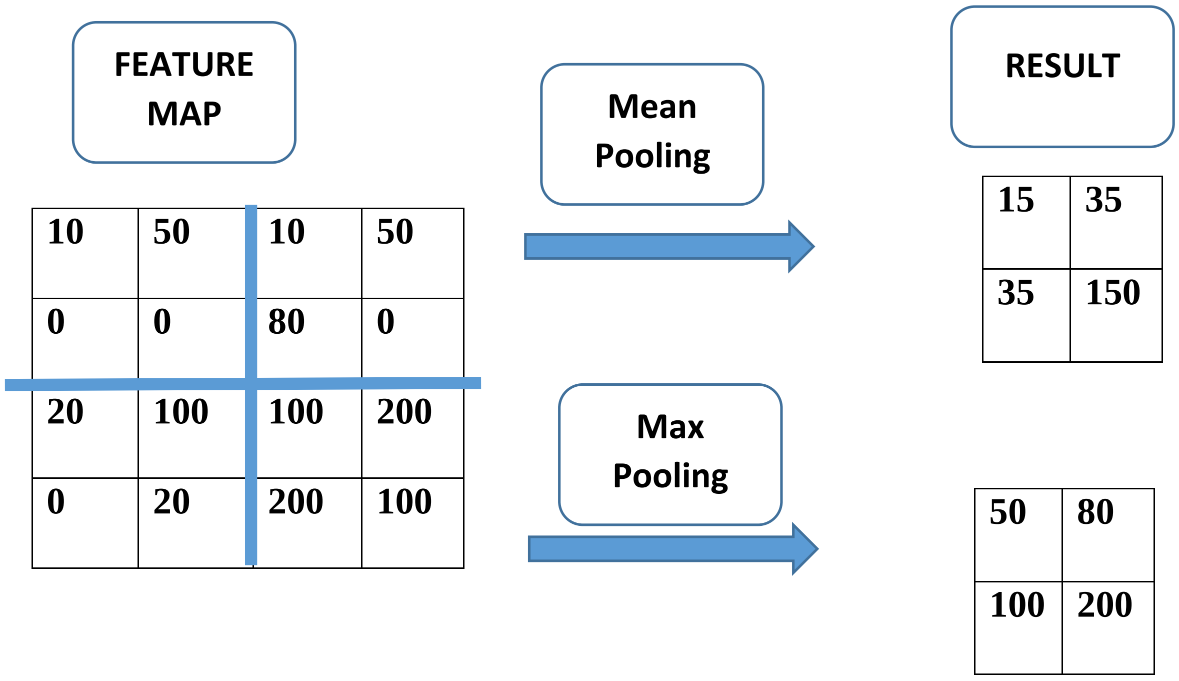


POOLING LAYER:

As for the pooling layer, it reduces the size of the image , as it combines neighbouring pixels of a certain area of the image into a single representative value. The neighbouring pixels are usually selected from the square matrix. The representative value is usually set as the mean or maximum of the selected pixels. The effect of the pooling layer can not be recovered due to the reduction of resolution and the removal of pixels but its advantage is that the new image will have less pixels and the computational load and the problem of overfitting will be reduced.

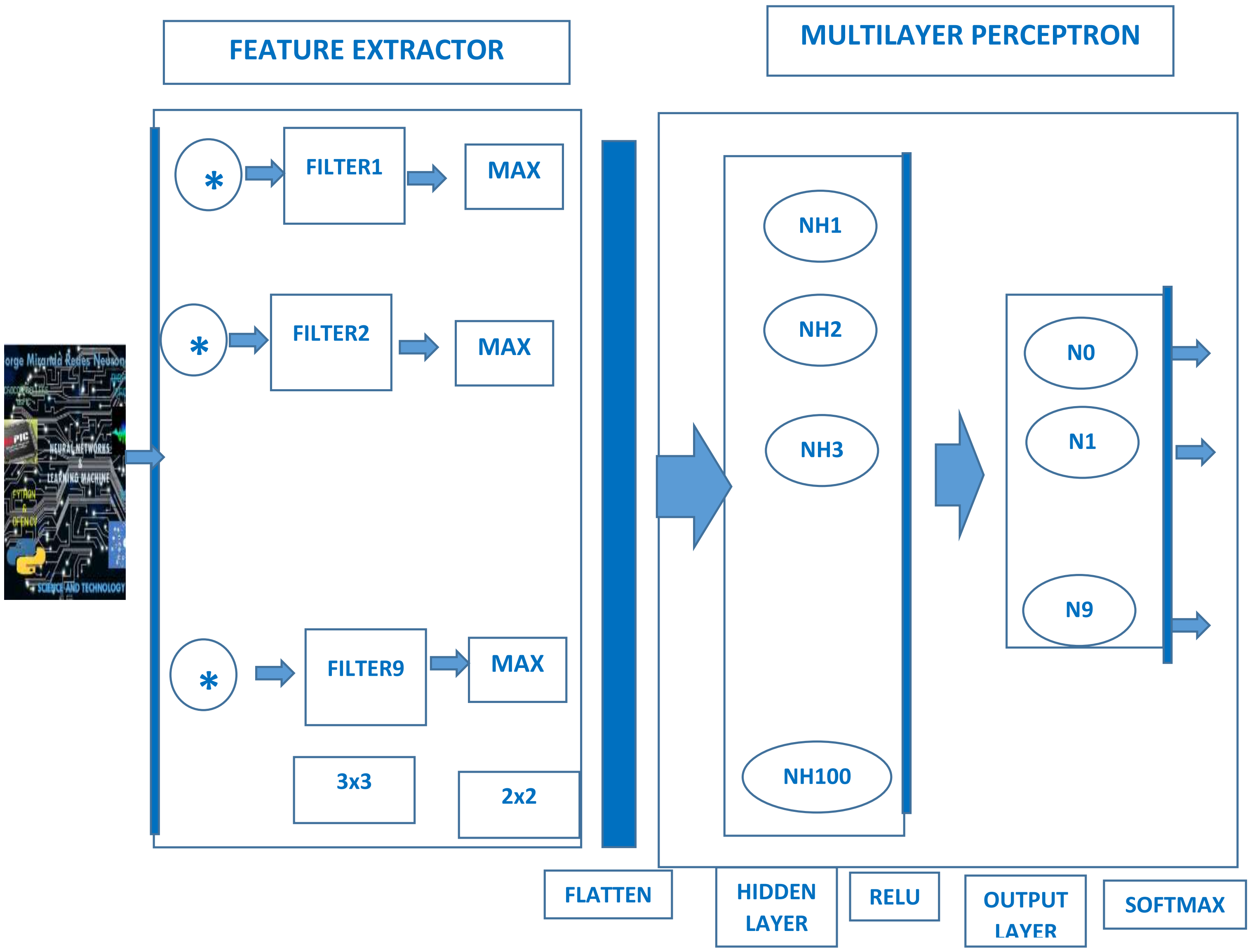
There two important choices to do the pooling operationg :

Mean pooling and Max pooling , take for example the following 2x2 pooling layer.



The operation of pooling is simple to handle. The feature map (image or matrix) passes through the pooling layer and it shrinks to a 2x2 pixel image. The resultant value of each pixel depends on the type of pooling layer (Mean Pooling or Max pooling).

PROPOSED ARCHITECTURE



The flatten step is used to vectorize the outputs of the convolution network or feature extractor.

FEATURE EXTRACTOR

N° of 2D filters=9,Activation function =linear, dimension=3x3

Dimension of pooling layer=2x2 , method = Max Pooling

CLASSIFIER

Hidden layer: 100 neurons , activation function = relu

Ouput layer:10 neurons , activation function = sofmax

KERAS :

According to the documentation of Keras , it is a High-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed in order to be used in a simple way thanks to the modules that it offers to be able to carry out research in the area of artificial intelligence.

- Runs seamlessly on CPU and GPU.
- Keras 2.2.5 was the last release to only support TensorFlow 1
- Keras 2.3.0 add support for TensorFow 2,0

The model is the core data structure of Keras that is a way to organize layers. The Sequential model represents a linear stack of layers.

Import numpy

```
import numpy as np
```

Import the Sequential Class from keras.models and then create the objet Sequential called CNN_model.

```
from keras.models import  Sequential

CNN_model=Sequential()
```

After we have already created our Sequential objet now convolutional layer , pooling layer and multilayer neural network must be added to the Sequential objet .

A way to add layers to the Sequential Object is by using the its **add** method.

Adding Convolutional Layer

We need to import the Conv2D class

```
from keras.layers import Conv2D

#number of filters  (9)
num_filters=9
```

```
#size of each filter (3x3)
filter_size=3
#dimension of input 28x28x1 (grayscale image)
shape_input=(28,28,1)

CNN_model.add(Conv2D(num_filters, filter_size,
shape_input))
```

Adding Pooling Layer:

We need to import the MaxPool2D class if we want to make use of the Max Pooling layer

```
from keras.layers import MaxPool2D

pool_size = 2

CNN_model.add(MaxPooling2D(pool_size=pool_size))
```

full code

```
import numpy as np
import mnist
from keras.models import  Sequential
from keras.layers import
Conv2D,MaxPooling2D,Dense,Flatten
from keras.utils import to_categorical
#load the data from the mnits data set
#training images (60000 records)
train_images=mnist.train_images()
train_labels=mnist.train_labels()
#testing images (10000 records)
test_images=mnist.test_images()
test_labels=mnist.test_labels()
#normalize the values to the range of [-0.5 0.5]
norm_train=train_images/255 -0.5
test_images=test_images/255 -0.5
norm_train = np.expand_dims(norm_train, axis=3)
test_images = np.expand_dims(test_images, axis=3)
#-------------------FEATURE EXTRACTOR--------

CNN_model=Sequential()
#number of filters  (9)
num_filters=9
#size of each filter (3x3)
filter_size=3
#dimension of input 28x28x1 (grayscale image)
shape_input=(28,28,1)
CNN_model.add(Conv2D(num_filters, filter_size,
input_shape=shape_input))
#size of the pooling layer (2x2)
size_pool=2
#max Polling layer
CNN_model.add(MaxPooling2D(pool_size=size_pool))
#add Flatten to the model
```

```
CNN_model.add(Flatten())


#-------------------CLASSIFIER NEURAL NETWORK------
#----------HIDDEN LAYER
#add a hidden layer  using the relu activation
function

CNN_model.add(Dense(100,activation="relu"))
#add a fully connected neural network using softmax
activation function
#---------OUTPUT LAYER
CNN_model.add(Dense(10,activation="softmax"))
#configure the model for the training of the CNN
#------------compile method------------------------
#algorithm
optimizer="adam"
#cost function
loss_="categorical_crossentropy"
#metric
metric=["accuracy"]

CNN_model.compile(optimizer,loss=loss_,metrics=metric)
#------------------------fit method---------------
--
#use of the fit method for the training of the CNN
#number of epochs
N_epochs=1
#number of data points for using in the batch
Batch_size=50

CNN_model.fit(norm_train,to_categorical(train_labels),epochs
=N_epochs,batch_size=Batch_size,shuffle=True,validation_data
=(test_images, to_categorical(test_labels)))

#save the weights in a hdf5 file to use in other
scripts
CNN_model.save_weights("pesos_w.h5")
```

```
#use of the testing data
predi=CNN_model.predict(test_images)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import
accuracy_score,precision_recall_fscore_support
#compute the confusion matrix
Matrix=confusion_matrix(test_labels,np.argmax(predi
,axis=1))
print("CONFUSION MATRIX Nclasses x Nclasses ")
print(Matrix)
#accuracy
accuracy=accuracy_score(test_labels,np.argmax(predi
,axis=1))
print("ACCURACY ",accuracy)
```

BE HAPPY, YOU NEVER KNOW HOW MUCH TIME YOU HAVE LEFT.