UNIVERSIDAD POLITÉCNICA
SALESIANA
ECUADOR

Project IA2

Name : Ruben Esteban Abad Ordoñez.Subject: I.A.2

Libreries.

In [1]:

```
import numpy as np
import os
import re
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
import keras
from keras.utils import to_categorical
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
```

```
Using TensorFlow backend.
```

In this Section right here, is where we fix and read the dataset

In [2]:

```
dirname = os.path.join(os.getcwd(), 'corpus')
imgpath = dirname + os.sep

images = []
directories = []
dircount = []
prevRoot=''
cant=0

print("leyendo imagenes de ",imgpath)

for root, dirnames, filenames in os.walk(imgpath):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant=cant+1
            filepath = os.path.join(root, filename)
            imag = plt.imread(filepath)
            imag = np.expand_dims(imag, axis=2)
            images.append(imag)

            b = "Leyendo..." + str(cant)
            print (b, end="\r")
            if prevRoot !=root:
                print(root, cant)
                prevRoot=root
                directories.append(root)
                dircount.append(cant)
                cant=0
dircount.append(cant)

dircount = dircount[1:]
dircount[0]=dircount[0]+1
print('Directorios leidos:',len(directories))
print("Imagenes en cada directorio", dircount)
print('suma Total de imagenes en subdirs:',sum(dircount))
```

```
leyendo imagenes de  C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_10_yna 1
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_11_taamatar 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_12_thaa 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_13_daa 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_14_dhaa 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_15_adna 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_16_tabala 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_17_tha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_18_da 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_19_dha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_1_ka 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_20_na 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_21_pa 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_22_pha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_23_ba 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_24_bha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_25_ma 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_26_yaw 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_27_ra 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_28_la 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_29_waw 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_2_kha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_30_motosaw 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_31_petchiryakha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_32_patalosaw 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_33_ha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_34_chhya 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_35_tra 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_36_gya 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_3_ga 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_4_gha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_5_kna 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_6_cha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_7_chha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_8_ja 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_9_jha 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_0 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_1 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_2 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_3 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_4 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_5 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_6 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_7 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_8 1700
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\digit_9 1700
Directorios leidos: 46
Imagenes en cada directorio [1701, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1
700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1
700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1
700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1699]
suma Total de imagenes en subdirs: 78200
```

I am going to obtain the labels, as these are the dataframe in x and y values.

In [3]:

```
labels=[]
indice=0
for cantidad in dircount:
    for i in range(cantidad):
        labels.append(indice)
    indice=indice+1
print("Cantidad etiquetas creadas: ",len(labels))

delta=[]
indice=0
for directorio in directories:
    name = directorio.split(os.sep)
    print(indice , name[len(name)-1])
    delta.append(name[len(name)-1])
    indice=indice+1

y = np.array(labels)
X = np.array(images, dtype=np.uint32) #turn into to list  numpy


# Find the unique numbers from the train labels
classes = np.unique(y)
nClasses = len(classes)
# print('Total number of outputs : ', nClasses)
print('Output classes : ', classes)
print('tamanio de y',len(y))
```

```
Cantidad etiquetas creadas:  78200
0 character_10_yna
1 character_11_taamatar
2 character_12_thaa
3 character_13_daa
4 character_14_dhaa
5 character_15_adna
6 character_16_tabala
7 character_17_tha
8 character_18_da
9 character_19_dha
10 character_1_ka
11 character_20_na
12 character_21_pa
13 character_22_pha
14 character_23_ba
15 character_24_bha
16 character_25_ma
17 character_26_yaw
18 character_27_ra
19 character_28_la
20 character_29_waw
21 character_2_kha
22 character_30_motosaw
23 character_31_petchiryakha
24 character_32_patalosaw
25 character_33_ha
26 character_34_chhya
27 character_35_tra
28 character_36_gya
29 character_3_ga
30 character_4_gha
31 character_5_kna
32 character_6_cha
33 character_7_chha
34 character_8_ja
35 character_9_jha
36 digit_0
37 digit_1
38 digit_2
39 digit_3
40 digit_4
41 digit_5
42 digit_6
43 digit_7
44 digit_8
45 digit_9
Output classes :  [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45]
tamanio de y 78200
```

In this part we apply the hotencoding to the respective outputs to the y values and also divide or separate in the data for our train and test.

In [4]:

```
Y_one_hot = to_categorical(y)
```

In [5]:

```
#print(len(images),'---',len(Y_one_hot))
train_X,test_X,train_Y,test_Y = train_test_split(X,Y_one_hot,test_size=0.2)
print(len(train_X))
print(len(test_X))
```

```
62560
15640
```

Now we proceed to the design of the neural network which consists of a hidden netra layer 3 and an y-output repectively will deal with alpha and an acceptable error for its use and increase the precision.

In [8]:

```
INIT_LR = 2e-2
epochs = 25
batch_size = 128
modelo = Sequential()
modelo.add(Dense(32, activation='relu',input_shape=(32,32,1)))
modelo.add(Dense(64, activation='relu'))
modelo.add(Dense(64, activation='relu'))
# modelo.add(Conv2D(32, kernel_size=(3, 3),activation='linear',padding='same',input_shape=(32,32,1)))
modelo.add(LeakyReLU(alpha=0.4))
modelo.add(MaxPooling2D((2, 2),padding='same'))
modelo.add(Dropout(0.4))

modelo.add(Flatten())
modelo.add(Dense(128, activation='relu'))
# modelo.add(Dense(32, activation='linear'))
modelo.add(LeakyReLU(alpha=0.4))
modelo.add(Dropout(0.3))
modelo.add(Dense(nClasses, activation='softmax'))
modelo.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
# modelo.summary()
# modelo.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adagrad(lr=INIT_LR, decay=INIT_LR / 100),metrics=['accuracy'])
```

This section includes training and its structure.

In this part is the training which took 45 min with 25 Epochs*

In [9]:

```
sport_train_dropout = modelo.fit(train_X, train_Y, batch_size=batch_size,epochs=
epochs,verbose=1,validation_data=(test_X, test_Y))
```

```
Train on 62560 samples, validate on 15640 samples
Epoch 1/25
62560/62560 [==============================] - 112s 2ms/step - loss: 1.8424 -
accuracy: 0.5017 - val_loss: 1.2196 - val_accuracy: 0.6666
Epoch 2/25
62560/62560 [==============================] - 120s 2ms/step - loss: 1.3390 -
accuracy: 0.6253 - val_loss: 1.0104 - val_accuracy: 0.7205
Epoch 3/25
62560/62560 [==============================] - 112s 2ms/step - loss: 1.2110 -
accuracy: 0.6571 - val_loss: 0.9226 - val_accuracy: 0.7471
Epoch 4/25
62560/62560 [==============================] - 116s 2ms/step - loss: 1.1301 -
accuracy: 0.6795 - val_loss: 0.8921 - val_accuracy: 0.7545
Epoch 5/25
62560/62560 [==============================] - 112s 2ms/step - loss: 1.0806 -
accuracy: 0.6901 - val_loss: 0.8518 - val_accuracy: 0.7676
Epoch 6/25
62560/62560 [==============================] - 113s 2ms/step - loss: 1.0394 -
accuracy: 0.7019 - val_loss: 0.8235 - val_accuracy: 0.7729
Epoch 7/25
62560/62560 [==============================] - 114s 2ms/step - loss: 1.0127 -
accuracy: 0.7095 - val_loss: 0.7951 - val_accuracy: 0.7834
Epoch 8/25
62560/62560 [==============================] - 113s 2ms/step - loss: 0.9848 -
accuracy: 0.7167 - val_loss: 0.7874 - val_accuracy: 0.7825
Epoch 9/25
62560/62560 [==============================] - 115s 2ms/step - loss: 0.9642 -
accuracy: 0.7212 - val_loss: 0.7680 - val_accuracy: 0.7857
Epoch 10/25
62560/62560 [==============================] - 114s 2ms/step - loss: 0.9473 -
accuracy: 0.7243 - val_loss: 0.7530 - val_accuracy: 0.7908
Epoch 11/25
62560/62560 [==============================] - 113s 2ms/step - loss: 0.9347 -
accuracy: 0.7279 - val_loss: 0.7358 - val_accuracy: 0.7952
Epoch 12/25
62560/62560 [==============================] - 113s 2ms/step - loss: 0.9226 -
accuracy: 0.7322 - val_loss: 0.7353 - val_accuracy: 0.7952
Epoch 13/25
62560/62560 [==============================] - 108s 2ms/step - loss: 0.9118 -
accuracy: 0.7358 - val_loss: 0.7277 - val_accuracy: 0.7971
Epoch 14/25
62560/62560 [==============================] - 109s 2ms/step - loss: 0.8952 -
accuracy: 0.7393 - val_loss: 0.7216 - val_accuracy: 0.8008
Epoch 15/25
62560/62560 [==============================] - 110s 2ms/step - loss: 0.8923 -
accuracy: 0.7387 - val_loss: 0.7208 - val_accuracy: 0.8024
Epoch 16/25
62560/62560 [==============================] - 110s 2ms/step - loss: 0.8828 -
accuracy: 0.7422 - val_loss: 0.7213 - val_accuracy: 0.8001
Epoch 17/25
62560/62560 [==============================] - 114s 2ms/step - loss: 0.8710 -
accuracy: 0.7464 - val_loss: 0.7181 - val_accuracy: 0.8049
Epoch 18/25
62560/62560 [==============================] - 115s 2ms/step - loss: 0.8689 -
accuracy: 0.7471 - val_loss: 0.7095 - val_accuracy: 0.8064
Epoch 19/25
62560/62560 [==============================] - 114s 2ms/step - loss: 0.8643 -
accuracy: 0.7472 - val_loss: 0.7012 - val_accuracy: 0.8072
Epoch 20/25
62560/62560 [==============================] - 114s 2ms/step - loss: 0.8595 -
accuracy: 0.7496 - val_loss: 0.7018 - val_accuracy: 0.8073
Epoch 21/25
62560/62560 [==============================] - 118s 2ms/step - loss: 0.8555 -
accuracy: 0.7487 - val_loss: 0.6977 - val_accuracy: 0.8081
Epoch 22/25
62560/62560 [==============================] - 114s 2ms/step - loss: 0.8491 -
accuracy: 0.7519 - val_loss: 0.6939 - val_accuracy: 0.8089
Epoch 23/25
62560/62560 [==============================] - 111s 2ms/step - loss: 0.8373 -
```

In this part is to estimate a % Precision

In [10]:

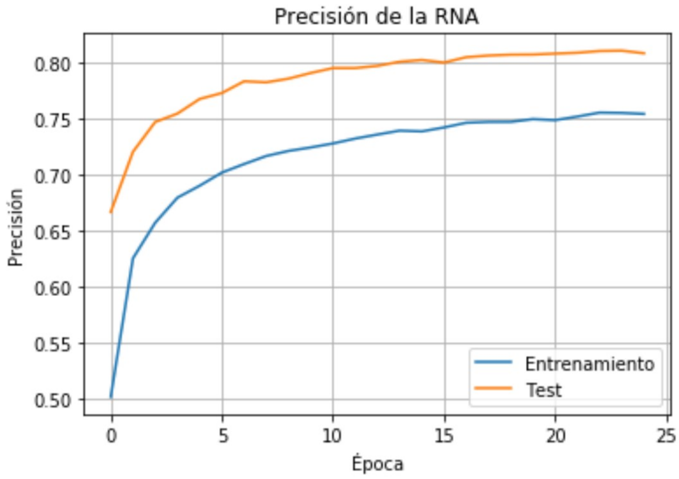
```
from sklearn.metrics import accuracy_score
y_pred = modelo.predict(test_X)

pred = list()
for i in range(len(y_pred)):
    pred.append(np.argmax(y_pred[i]))

test = list()
for i in range(len(test_Y)):
    test.append(np.argmax(test_Y[i]))


precision = accuracy_score(pred, test) # Comparamos lo que predijo la red con las salidas deseadas
print('Precisión: ',(precision*100))
```

```
Precisión:  80.84398976982096
```

In this part is to draw about to train and test

In [11]:

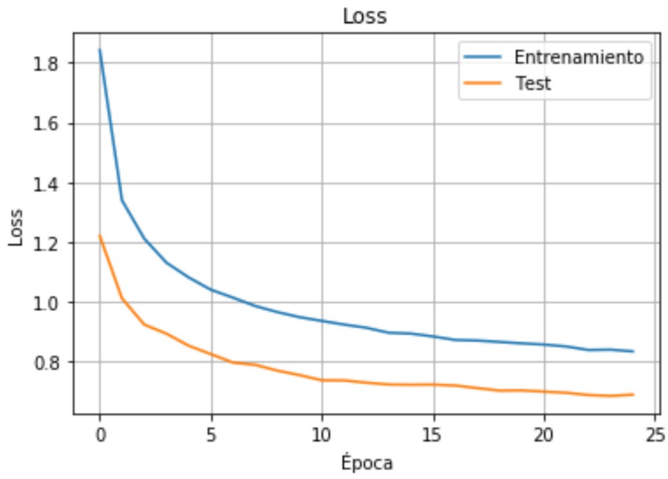
```
import matplotlib.pyplot as pp

%matplotlib inline

#print(historial.history)

pp.plot(sport_train_dropout.history['accuracy'])
pp.plot(sport_train_dropout.history['val_accuracy'])
pp.title('Precisión de la RNA')
pp.ylabel('Precisión')
pp.xlabel('Época')
pp.legend(['Entrenamiento','Test'], loc='lower right')
pp.grid(True)
pp.show()
```



In this part is to draw about to loss witnh respective to Epoch

In [12]:

```
pp.plot(sport_train_dropout.history['loss'])
pp.plot(sport_train_dropout.history['val_loss'])
pp.title('Loss')
pp.ylabel('Loss')
pp.xlabel('Época')
pp.legend(['Entrenamiento','Test'], loc='upper right')
pp.grid(True)
pp.show()
```



In this part is to save the model .

In [13]:

```
modelo.save_weights("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

In this part is doing to Prediction with Image

In [14]:

```
import tensorflow as tf
from PIL import Image
dirnam = os.path.join(os.getcwd(),)
print(dirnam)
image = Image.open("1380.png")
```

```
C:\Users\Ruben\OneDrive\UPS\10mo\IA2
```

In [15]:

```
input_arr = keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])  # Convert single image to a batch.
predictions = modelo.predict(input_arr)
print(len(predictions))
```

```
1
```

In [16]:

```
print("Imágen:{}".format(predictions))
```

```
Imágen:[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Finally the prediction procedure goes with which it is established as a parameter to the image itself for its prediction and launches its respective output.

In [17]:

```
for i in range(len(predictions)):
    for j in range (len(predictions[i])):
        if(predictions[i][j]!=0):
            print((predictions[i][j]))
            print(j)
            print(directories[j])
            name = directories[j].split(os.sep)
            print(name[len(name)-1])
            break
```

```
1.0
10
C:\Users\Ruben\OneDrive\UPS\10mo\IA2\corpus\character_1_ka
character_1_ka
```