In [ ]:
```
# 1-Libraries

import cv2
import math
import numpy as np
import pandas as pd
from glob import glob

import matplotlib.pyplot as plt
%matplotlib inline

from sklearn import preprocessing
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
```

In [ ]:
```
# 2-Data loading

train_path ='/content/drive/My Drive/CNN_project/train.zip'
!mkdir temp_train

from zipfile import ZipFile
with ZipFile ( train_path, 'r') as zip:
    zip.extractall('/content/drive/My Drive/CNN_project/temp_train')

#-----------------------------------------------------------------------------

# The upper Zip extraction wasn't working no matter what !!
# I even uploaded the folder of unzipped pics 'train' and replaced the lower path as:

# path_to_images ='content/drive/My Drive/CNN_project/train/*/*.png'
# But still to no avail !  :-(

# THE CODE from here down is to the best of my guessing on how this CNN project should evolve.
# It was done on my desktop as I lost hope with reading the images into to the code :-(

#-----------------------------------------------------------------------------
path_to_images ='content/drive/My Drive/CNN_project/temp_train/*/*.png'
images = glob(path_to_images)

trainImg   = []
trainLabel = []

count=1
num = len(images)
for img in images:
    print(str(j) + '/' + str(num) , end= '\r' )
    trainImg.append ( cv2.resize(cv2.imread(img),(128,128)) )
    trainLabel.append (img.split('/')[-2])
    count+=1

trainImg   = np.asarray(trainImg)
trainLabel = pd.DataFrame(trainLabel)

print(trainImg.shape)
print(trainLabel.shape)
```

In [ ]:
```
# 3- Show some Original images

for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(trainImg[i])
```

In [ ]:
```
# 4 - Blur the images

blurred_trainImg = []

for img in trainImg:
    blurImg = cv2.GaussianBlur(img ,(5,5), 0)
    blurred_trainImg.append(blurImg)


for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(blurred_trainImg[i])
```

In [ ]:
```
# 5 - Normalising the blurred training set

blurred_trainImg = blurred_trainImg /255.0
```

In [ ]:
```
 # 7- One hot encoding of the Labels set

labels = preprocessing.LabelEncoder()
labels.fit(trainLabel[0])
print('Classes' + str(labels.classes))

encodedlabels = labels.transform(trainlabel[0])
clearalllabels = to_categorically(encodedlabels )
classes = clearalllabels.shape[1]

print(str(classes))
trainlabel[0].value_counts.plot(kind= 'bar')
```

In [ ]:
```
# 8- X and y

X = blurred_trainImg
y = clearalllabels
```

In [ ]:
```
# 9- train , validation , test sets : 70% , 15%, 15%

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size = 0.3, random_state=0, str
atify = y)

X_val, X_test, y_val, y_test     = train_test_split(X_temp, y_temp, test_size = 0.5, random_st
ate=0, stratify = y_temp)
```

In [ ]:
```
# 10- CNN Architecture

model = Sequential()

# Input layer
model.add(Conv2D(filters=64, kernel_size=(5, 5), input_shape=(128, 128, 3), activation='relu')
)

# 3 Convolution layers
model.add(Conv2D(filters=64, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))


model.add(Conv2D(filters=128, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))


model.add(Conv2D(filters=256, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.1))

# 2 fully connected layers
model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# One output layer
model.add(Dense(12, activation='softmax'))

model.summary()



# compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [ ]:
```
# 11 - Fitting the model

model.fit( x=X_train, y=y_train,
          batch_size=32,
          epochs=15 ,
          validation_data=(X_val, y_val),
          shuffle=True,
          verbose=1)
```

In [ ]:
```
#12 - Model Evaluation

scores = model.evaluate ( X_test , y_test, verbose=1)
print( 'Test loss: ', score[0])
print( 'Test accuracy: ', score[1])
```

In [ ]:
```
# 13- Model Prediction  and Confusion matrix
y_pred = model.predict(X_test)

print('===Confusion matrix===')
cm = confusion_matrix(y_test, y_pred)
print(cm)


print('===Classification report===')
print(classification_report(y_test, y_pred))
```

In [ ]:
```
# 14- Visualise some predictions

for i in [2,3,33,36,59]:
    plt.imshow(X_test[i].reshape(128,128,3))
    y_pred = model.predict(X_test[i], reshape(1,128,128,3))
    print('Predicted label: ', y_pred.argmax())
```