INFORME DE DETECCIÓN DE EMOCIONES FACIALES

Elena Pérez, Miguel Muzo
Escuela de Formación de Tecnólogos
Escuela Politécnica Nacional
Quito, Ecuador
ligia.perez@epn.edu.ec, eduardo.muzo@epn.edu.ec

***Abstract*— El rol del reconocimiento automático de emociones está creciendo de forma continua actualmente. A medida que los computadores se vuelven más y más sofisticados, ya sea a nivel profesional o social, se vuelve más importante que estas sean capaces de interactuar de forma natural. Se considera que las máquinas deben incluir este tipo de inteligencia de forma de reconocer el estado afectivo dado ciertas señales psicológicas.**

I. INTRODUCCIÓN

El rol del reconocimiento automático de emociones está creciendo de forma continua actualmente. Esto se debe a que se ha aceptado la importancia que tiene la reacción a los estados afectivos del usuario en la interacción persona-computador. A medida que los computadores se vuelven más y más sofisticadas, ya sea a nivel profesional o social, se vuelve más importante que estas sean capaz de interactuar de forma natural, o sea, de forma similar a como se interactúa con otros agentes humanos. La característica más importante de la interacción humana que garantiza que el proceso se haga de forma natural, es el proceso por el cual podemos inferir el estado emocional de otros.

II. HERRAMIENTAS UTILIZADAS

- KERAS

Es una librería de código abierto (con licencia MIT) escrita en Python para acelerar la creación de redes neuronales para ello, Keras no funciona como un framework independiente, sino como una interfaz de uso intuitivo (API) que permite acceder a varios frameworks de aprendizaje automático y desarrollarlos.

¿Cómo funciona KERAS?

Proporciona bloques modulares sobre los que se pueden desarrollar modelos complejos de aprendizaje profundo. A diferencia de los frameworks, este software de código abierto no se utiliza para operaciones sencillas de bajo nivel, sino que utiliza las bibliotecas de los frameworks de aprendizaje automático vinculadas, que en cierto modo actúan como un motor de backend para Keras. Fig. 1

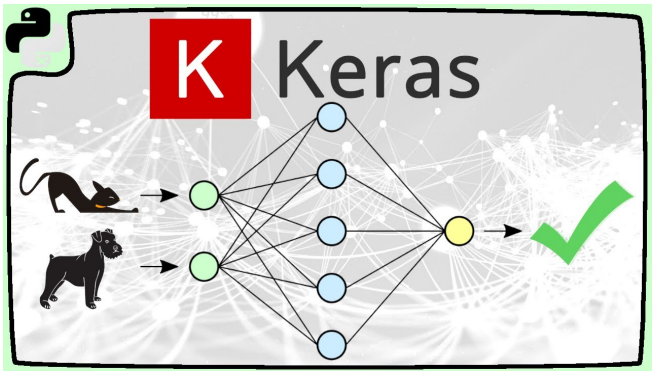

Fig. 1 Keras su funcinamiento.

- MATPLOTLIB

Es una librería de Python especializada en la creación de gráficos en dos dimensiones, también a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, pylab, diseñada para recordar a la de MATLAB. Matplotlib se basa en varios elementos clave. Una “figura” es una ilustración completa. Cada trazado de esa figura se llama “eje”.

El “plotting” consiste en crear una gráfica. Es necesario utilizar datos, en forma de pares clave/valor que constituyen

los ejes X e Y. Después se utilizan funciones como “scatter”, “bar” y “pie” para crear el esquema.

Es posible generar gráficas básicas como diagramas de barras o histogramas, pero también figuras más complejas en tres dimensiones. Fig. 2

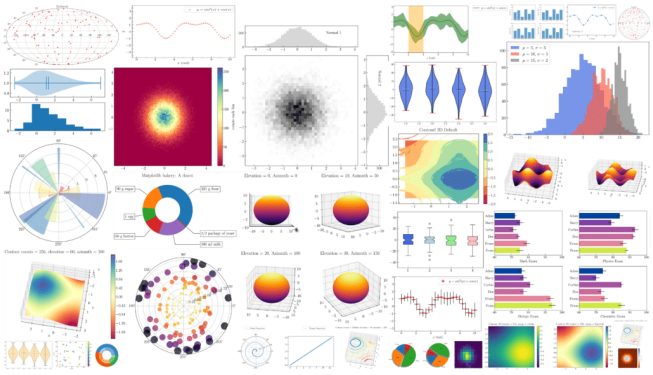

Fig. 2 Graficos de matplotlib

- SKLEARN

Es una de estas librerías gratuitas para Python. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad. Además, presenta la compatibilidad con otras librerías de Python como NumPy, SciPy y matplotlib, es la librería más útil para Machine Learning en Python, es de código abierto y es reutilizable en varios contextos, fomentando el uso académico y comercial. Proporciona una gama de algoritmos de aprendizaje supervisados y no supervisados en Python. Fig 3

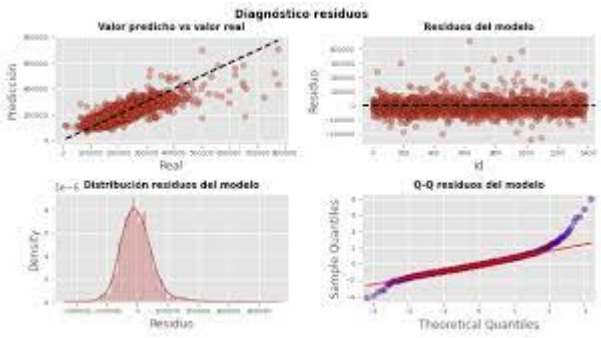

Fig. 3. Resultados de agrupación con sklearn.

- VISUAL STUDIO

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

VS Code tiene una gran variedad de características útiles para agilizar el trabajo, que lo hacen el editor preferido por muchos (me incluyo) para trabajar los proyectos.

- Multiplataforma: Es una característica importante en cualquier aplicación y más si se trata de desarrollo. Visual Studio Code está disponible para Windows, GNU/Linux y macOS.
- IntelliSense: Esta característica está relacionada con la edición de código, autocompletado y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código.
- Depuración: Visual Studio Code incluye la función de depuración que ayuda a detectar errores en el código.
- Uso del control de versiones: Visual Studio Code tiene compatibilidad con Git, por lo que puedes revisar diferencias o lo que conocemos con git diff, organizar archivos, realizar commits desde el editor, y hacer push y pull desde cualquier servicio de gestión de código fuente (SMC).
- Extensiones: Visual Studio Code es un editor potente y en gran parte por las extensiones. Las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada. Fig 4

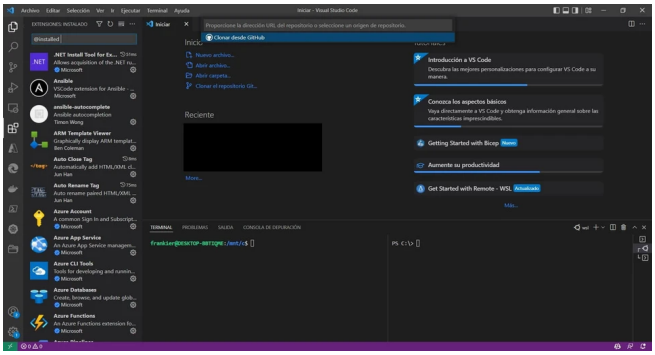

Fig. 4. Interfaz de Visual Studio Code.

III. DESARROLLO DEL PROYECTO

Para realizar el proyecto se requirió la base de datos FER-2013 de la plataforma de Kaggle, es importante almacenar dentro de la carpeta donde se encuentra el código del proyecto.

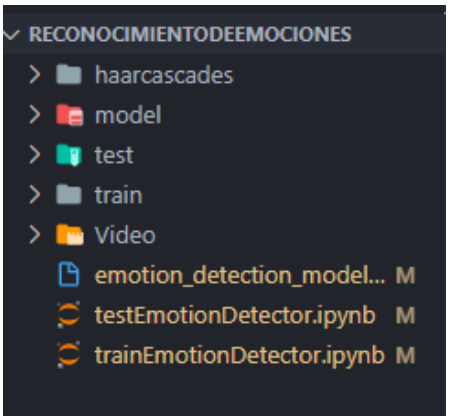

Fig. 5 Carpeta Reconocimiento de emociones.

Se instaló previamente las librerías de keras, matpltlib, numpy, tensorflow, ya que se utilizara para poder realizar el proyecto, luego se llamó a las librerías dentro del código del proyecto.

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D
import os
from matplotlib import pyplot as plt
import numpy as np
```

Fig. 6 Librerías del proyecto.

Después se crea un tamaño para las imagenes que se van a mostrar dentro del programa, y para poder leer la dataset se crean las variables.

```
#image size
IMG_HEIGHT=48
IMG_WIDTH = 48
batch_size=32


train_data_dir='./train/'
validation_data_dir='./test/'
```

Fig. 7 Variables para el dataset.

Se utiliza la librería de keras, la cual cuenta con la clase ImageDataGenerator, la cual va ayudar a generar bloques o batch, esto se realiza porque cuando se requiere entrenar modelos con resoluciones mayores, por lo que se requiere mucha memoria, por lo tanto se necesita dividir el proceso de entrenamiento en bloques de menor tamaño de imágenes, por eso se utiliza la librería keras, en la que se realiza la técnica llamada data augmentation.

```
train_datagen = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=30,
                    shear_range=0.3,
                    zoom_range=0.3,
                    horizontal_flip=True,
                    fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
                    train_data_dir,
                    color_mode='grayscale',
                    target_size=(IMG_HEIGHT, IMG_WIDTH),
                    batch_size=batch_size,
                    class_mode='categorical',
                    shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
                            validation_data_dir,
                            color_mode='grayscale',
                            target_size=(IMG_HEIGHT, IMG_WIDTH),
                            batch_size=batch_size,
                            class_mode='categorical',
                            shuffle=True)
```

Fig. 8 Modelo con ImageDataGenerator.

Se verifica el generador trazando las emociones faciales de las imágenes generadas aleatoriamente, para ello se crea un arreglo denominado class_labels, en la que se debe incluir, las 7 emociones, por ejemplo: feliz, triste, nervioso, enojado, neutral, sorprendido y finalmente se imprime una función random las imágenes.

```
class_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']

img, label = train_generator.__next__()

import random

i=random.randint(0, (img.shape[0])-1)
image = img[i]
labl = class_labels[label[i].argmax()]
plt.imshow(image[:,:,0], cmap='gray')
plt.title(labl)
plt.show()
```

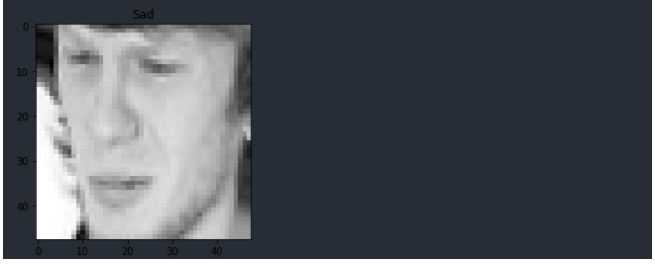

Fig. 9 Generador de emociones faciales.

Se crea un modelo para que una simple pila de capas donde cada capa va a tener exactamente un tensor de entrada y salida, por ello es importante especificar la forma de entrada, generalmente todas las capas en keras necesitan conocer la forma de las entrada para poder crear los pesos, cuando se crea una una capa inicialmente no tiene peso, una

vez que se haya creado el modelo se llamó a la función summary() es un método para poder mostrar todo el contenido del modelo.

```
# Create the model
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(7, activation='softmax'))

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

Fig. 10 Creación de modelos.

Para que el algoritmo pueda ser entrenado se crea una variable, el cual va a funcionar con los datos de entrenamiento, para eso se asigna un número aleatorio o predefinido por el usuario dentro de la variable espochs.

```
epochs=3

history=model.fit(train_generator,
                  steps_per_epoch=num_train_imgs//batch_size,
                  epochs=epochs,
                  validation_data=validation_generator,
                  validation_steps=num_test_imgs//batch_size)

model.save('emotion_detection_model_100epochs.h5')
```

Fig. 11 Variable espochs.

Se crean variables para visualizar en la gráfica, las variables de accuary y val accuary, después de crear la gráfica, con la ayuda de la librería matploitb.

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Fig. 12. Diagrama trading loss and validation loss.

Finalmente se construye, para la matriz confusión con ayuda de la librería tensorflow con el metodo sklearn.metrics.

```
#Confusion Matrix - verify accuracy of each class
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(test_labels, predictions)
print(cm)
import seaborn as sns
sns.heatmap(cm, annot=True)
```

```
[[2 2 1 0 0 0]
 [0 0 1 0 0 1]
 [0 0 7 0 2 1]
 [1 1 3 1 1 1]
 [0 1 1 0 2 0]
 [0 0 0 0 0 3]]

<AxesSubplot:>
```

Fig. 13. Matriz Confusión.

Para la comprobación sobre el reconocimiento de emociones faciales se crea nuevamente a class labels en la que se incluyen todas las emociones dentro de un arreglo y se procede a imprimir aleatoriamente las imágenes.

```
class_labels=['Angry','Disgust', 'Fear', 'Happy','Neutral','Sad','Surprise']
#Check results on a few select images
n=random.randint(0, test_img.shape[0] - 1)
image = test_img[n]
orig_labl = class_labels[test_labels[n]]
pred_labl = class_labels[predictions[n]]
plt.imshow(image[:,:,0], cmap='gray')
plt.title("Original label is:"+orig_labl+" Predicted is: "+ pred_labl)
plt.show()
```

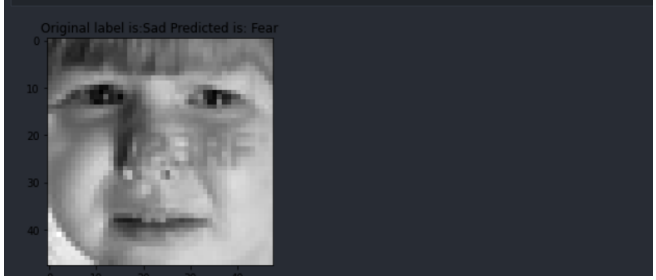

Fig. 14 Impresión de imágenes de la BD.

Para la comprobación de emociones mediante un video se debe instalar cv2, numpy y keras importante a model from json.

```
import cv2
import numpy as np
from keras.models import model_from_json
```

Fig. 15 Importación de librerías

Se crea un objeto en el que se incluyen todas las emociones dentro de la base de datos.

```
emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}
```

Fig. 16 Creación de objetos sobre la detección de emociones.

Se llama a al json que se encuentra en la carpeta, la cual sirve para reconocer las emociones dentro de un video.

```
# load json and create model
json_file = open('./model/emotion_model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
emotion_model = model_from_json(loaded_model_json)
```

Fig. 17 Creación de un modelo..

Se crea una variable denominada cap y se utiliza cv2 en la cual se incluye un método videoCapture que de igual manera se debe incluir dentro de la carpeta del proyecto.

```
cap = cv2.VideoCapture("./Video/video-01.mp4")
```

Fig. 18 Variable del video.

Crear la cascada para poder graficar la caja delimitadora que se va a encontrar alrededor de la cara, de igual manera se incluye funciones para poder detectar los rostros de las personas dentro del video.

```
while True:
    # Encuentra la cascada haar para dibujar la caja delimitadora alrededor de la cara
    ret, frame = cap.read()
    frame = cv2.resize(frame, (1280, 720))
    if not ret:
        break
    face_detector = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detectar los rostros disponibles en la cámara
    num_faces = face_detector.detectMultiScale(gray_frame, scaleFactor=1.3, minNeighbors=5)

    # tomar cada rostro disponible en la cámara y preprocesarlo
    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 4)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 0)

        # predict the emotions
        emotion_prediction = emotion_model.predict(cropped_img)
        maxindex = int(np.argmax(emotion_prediction))
        cv2.putText(frame, emotion_dict[maxindex], (x+5, y-20), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)

    cv2.imshow('Emotion Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Fig. 19 Crear la cascada para la detección de emociones.

Se reproduce el video y se comienza el programa a detectar las emociones faciales.

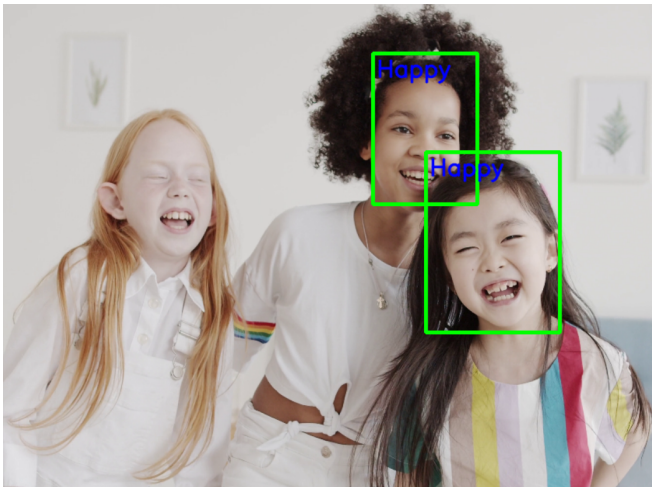

Fig. 20 Programa detectando emociones faciales.

IV. CONCLUSIONES

La inteligencia artificial en los últimos tiempo se a convertido muy fundamental para la vida diaria dentro de la sociedad, esto ayudando al avance tecnológico con la intención de tener mejores resultados, aunque es importante conocer que la IA no esta elaborada para reemplazar al hombre sino para ayudar, la detección de emociones se aplicado en varias aplicaciones, principalmente para las redes sociales.

V. REFERENCIAS

[1] Anonimo, «Digital Guide IONOS,» 08 Octubre 2020. [En línea]. Available: https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-keras/. [Último acceso: 31 Agosto 2022].

[2] José, «XC,» 04 Octubre 2020. [En línea]. Available: https://aprendeconalf.es/docencia/python/manual/matplotlib/. [Último acceso: 31 Agosto 2022].

[3] Anonimo, «Universidad Alcalá,» 15 Junio 2020. [En línea]. Available: https://www.master-data-scientist.com/scikit-learn-data-science/. [Último acceso: 31 Agosto 2022].

[4] F. Flores, «OpenWebinars,» 22 Julio 2022. [En línea]. Available: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/. [Último acceso: 31 Agosto 2022].

[5] Heiwitt, «Packard,» 16 Octubre 2020. [En línea]. Available: https://www.hpe.com/mx/es/what-is/artificial-intelligence.html?jumpid=ps_sgywi7pztj_aid-520061736&ef_id=CjwKCAjw9suYBhBIEiwA7iMhNOaH3wDXt35rZygDGPKg2OaU7z3hYNbURDPRsEpOgyCUTDgEMpz2txoCV40QAvD_BwE:G:s&s_kwcid=AL!13472!3!594978471597!e!!g!!qu%C3%A9%20es%20l. [Último acceso: 31 Agosto 2022].