CENTRO DE CIENCIAS BÁSICAS

Dpto. de Ciencias de la Computación.

Machine Learning - Deep Learning

Proyecto Final de Deep Learning

Orduña Salas Diego Antonio

Ingeniería en Computación Inteligente.

Francisco Javier Luna Rosas

15/05/2023

Evidencias

Recolección de datos

En esta fase se crea un notebook dataset_creation.ipynb y se consideraron las emociones `["angry", "happy", "sad", "surprise", "neutral"]` Para cada emoción se crea una carpeta dentro de la carpeta dataset:

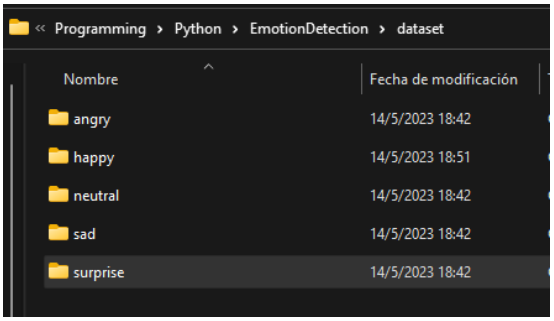
```
Labels = ["angry", "happy", "sad", "surprise", "neutral"]

for label in Labels:
    if not os.path.exists(label):
        os.mkdir(label)
```

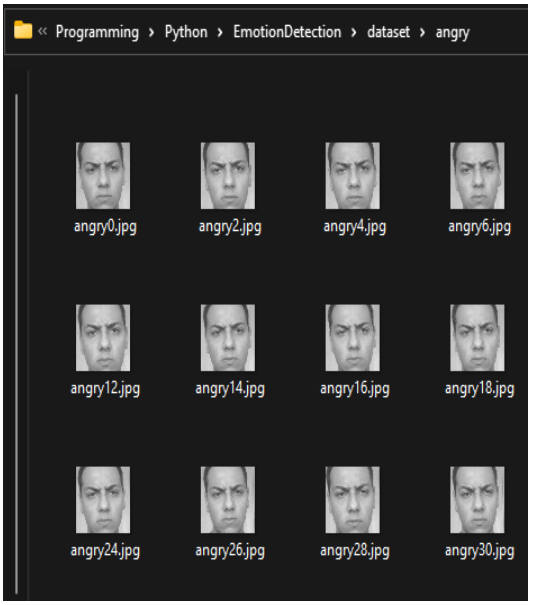
Finalmente para cada carpeta se captura cierto número de imágenes del rostro del sujeto con el siguiente bloque de código:

```
import cv2 as cv

for folder in Labels:
    #using count variable to name the images in the dataset.
    count = 0
    #Taking input to start the capturing
    print("Press 's' to start data collection for "+folder)
    userinput = input()
    if userinput != 's':
        print("Wrong Input.........")
        exit()
    #clicking 200 images per label, you could change as you want.
    while count<400:
        #read returns two values one is the exit code and other
is the frame
        status, frame = camera.read()
        #check if we get the frame or not
        if not status:
            print("Frame is not been captured..Exiting...")
            break
        #convert the image into gray format for fast caculation
```

```
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        #display window with gray image
        cv.imshow("Video Window",gray)

        #using cv to detect faces in the frame and croppe it.
this is what is going to be stored in the dataset
        face_cascade = cv.CascadeClassifier(cv.data.haarcascades
+ "haarcascade_frontalface_default.xml")
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for x,y,w,h in faces:
            cv.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 1)
            roi_gray = gray[y:y+h, x:x+w]
            cropped_img = cv.resize(roi_gray,(48,48))
            cv.normalize(cropped_img, cropped_img, alpha=0,
beta=1, norm_type=cv.NORM_L2, dtype=cv.CV_32F)
            cv.imwrite(folder+'/'+ folder
+str(count)+'.jpg',cropped_img)
            count+=1
        #display the frame with rectangle around the face
        cv.imshow("Video Window",frame)
        count+=1

        #to quite the display window press 'q'
        if cv.waitKey(1) & 0xFF == ord('q'):
            break

# When everything done, release the capture
camera.release()
cv.destroyAllWindows()
```

El resultado para cada subcarpeta de /dataset se ve de la siguiente forma:





Una vez implementado lo anterior, se hace un procesamiento en el cual se prepara el dataset para ser utilizado en una red neuronal, por lo que se hace la adaptación a los sentimientos, la vectorización de los píxeles y el uso que tendrá la imagen. Todo esto se realiza en data_transform.py

```
import pandas as pd
import numpy as np
import os
import cv2

# Definir las emociones y sus correspondientes valores enteros
emotion_dict = {0: "angry", 1: "happy", 2: "sad", 3: "surprise",
4: "neutral"}

# Crear listas vacías para almacenar los datos del conjunto de
datos
emotion = []
pixels = []
usage = []

# Iterar a través de cada carpeta de emociones
for emotion_folder in os.listdir("dataset"):
    # Obtener el valor entero correspondiente a la emoción actual
    emotion_label = list(emotion_dict.keys())[
```

```
        list(emotion_dict.values()).index(emotion_folder)
    ]

    # Iterar a través de cada imagen en la carpeta actual
    for image_filename in os.listdir(os.path.join("dataset",
emotion_folder)):
        # Leer la imagen y convertirla a escala de grises
        image_path = os.path.join("dataset", emotion_folder,
image_filename)
        image = cv2.imread(image_path)
        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        # Aplanar la imagen en un vector de una dimensión
        flat_image = gray_image.flatten()

        # Agregar los datos al conjunto de datos
        emotion.append(emotion_label)
        flat_image_to_string = " ".join(str(x) for x in
list(flat_image))
        pixels.append(flat_image_to_string)

        # Establecer la etiqueta de uso del conjunto de datos
(Training, PrivateTest, PublicTest)
        rand_num = np.random.rand()
        if rand_num < 0.5:
            usage.append("Training")
        elif rand_num < 0.75:
            usage.append("PrivateTest")
        else:
            usage.append("PublicTest")

# Crear un dataframe de pandas para almacenar los datos del
conjunto de datos
df = pd.DataFrame({"emotion": emotion, "pixels": pixels, "Usage":
usage})
```

```
# Guardar el conjunto de datos como un archivo CSV
df.to_csv("dataset.csv", index=False)
```

Con esto, podemos utilizar dataset.csv en nuestra red neuronal, la cual se construye en la siguiente fase

Red CNN

Para la implementación, cargamos el dataset:

```
df = pd.read_csv("dataset.csv")
df
```

✓ 0.1s

	emotion	pixels	Usage
0	0	169 171 163 67 59 74 72 96 145 160 166 174 183...	PublicTest
1	0	178 174 170 104 71 71 77 75 138 156 172 179 18...	Training
2	0	180 178 168 82 69 71 68 100 158 162 174 183 18...	PublicTest
3	0	182 175 175 89 71 72 72 88 152 161 170 177 180...	Training
4	0	180 176 179 78 73 76 77 100 157 167 174 177 18...	PublicTest
...	...	...	...
984	3	178 181 176 171 183 186 178 118 70 73 70 67 10...	PrivateTest
985	3	180 179 181 179 178 180 182 178 130 75 69 66 6...	Training
986	3	176 179 182 177 184 186 176 167 82 65 74 72 76...	Training
987	3	179 180 175 183 182 180 187 171 129 71 65 68 6...	PrivateTest
988	3	176 179 179 184 179 184 178 179 160 99 62 67 6...	Training

989 rows × 3 columns

Procedemos a implementar la arquitectura de la CNN

```
# Definir el modelo de la red neuronal convolucional
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(0.25))
```

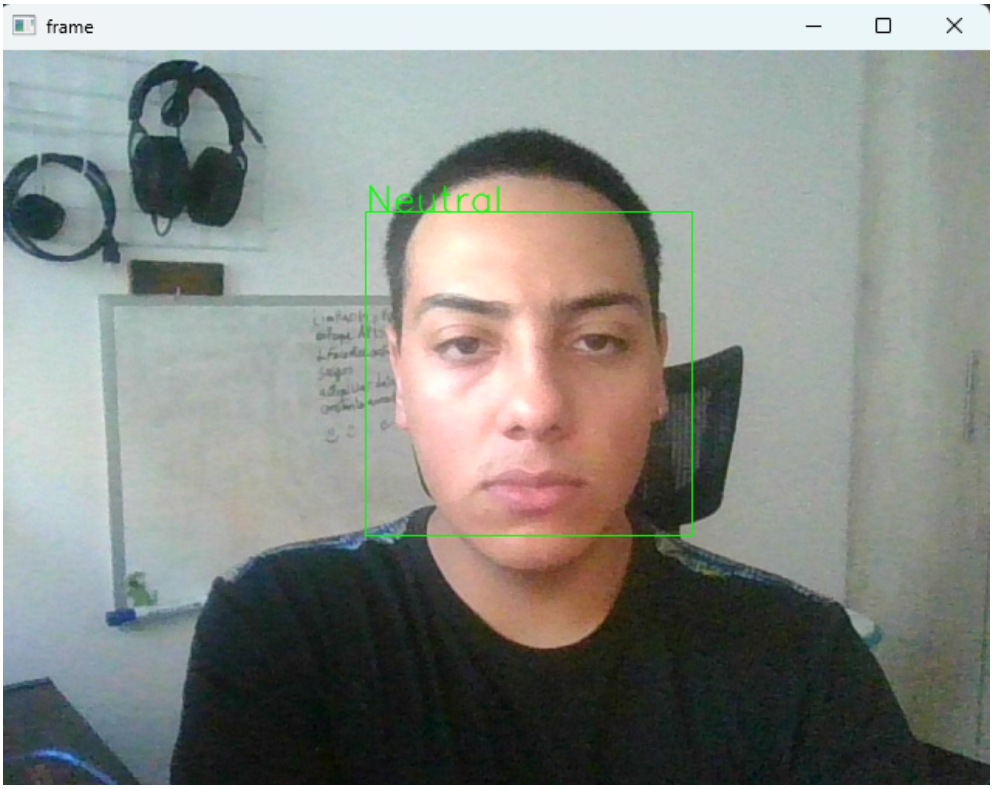
```python
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(5, activation="softmax"))

# Compilar el modelo
model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])
```

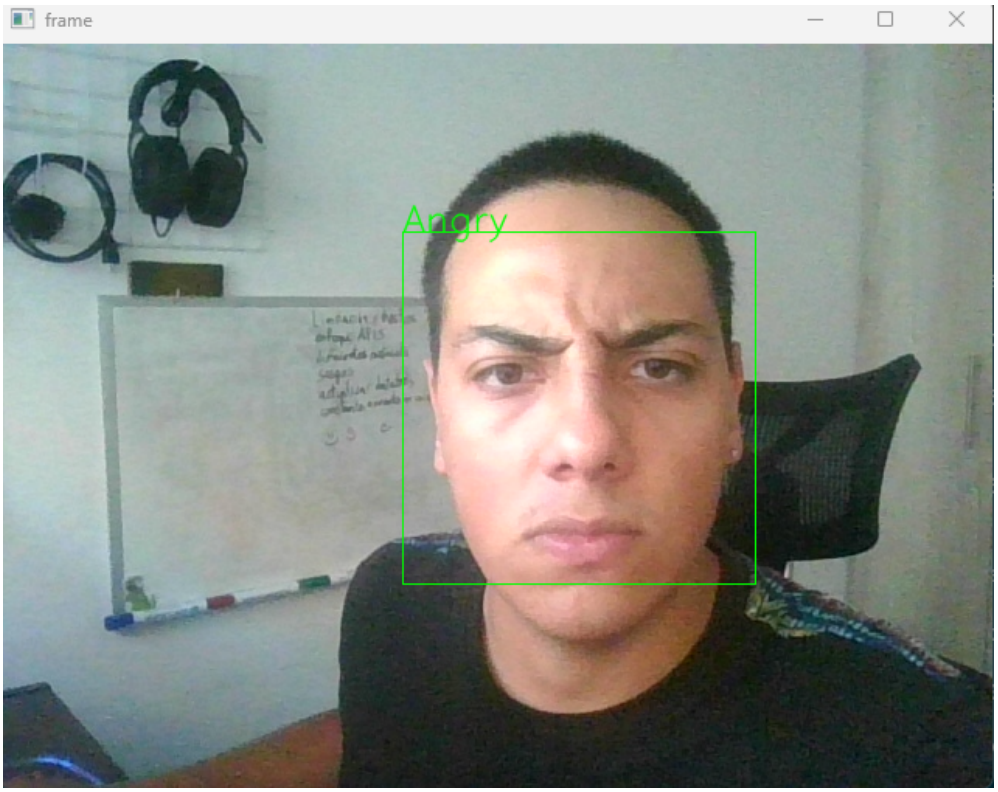
```
model.summary()
[6] ✓ 0.0s
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3965056
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 3,984,517
Trainable params: 3,984,517
Non-trainable params: 0
_________________________________________________________________
```

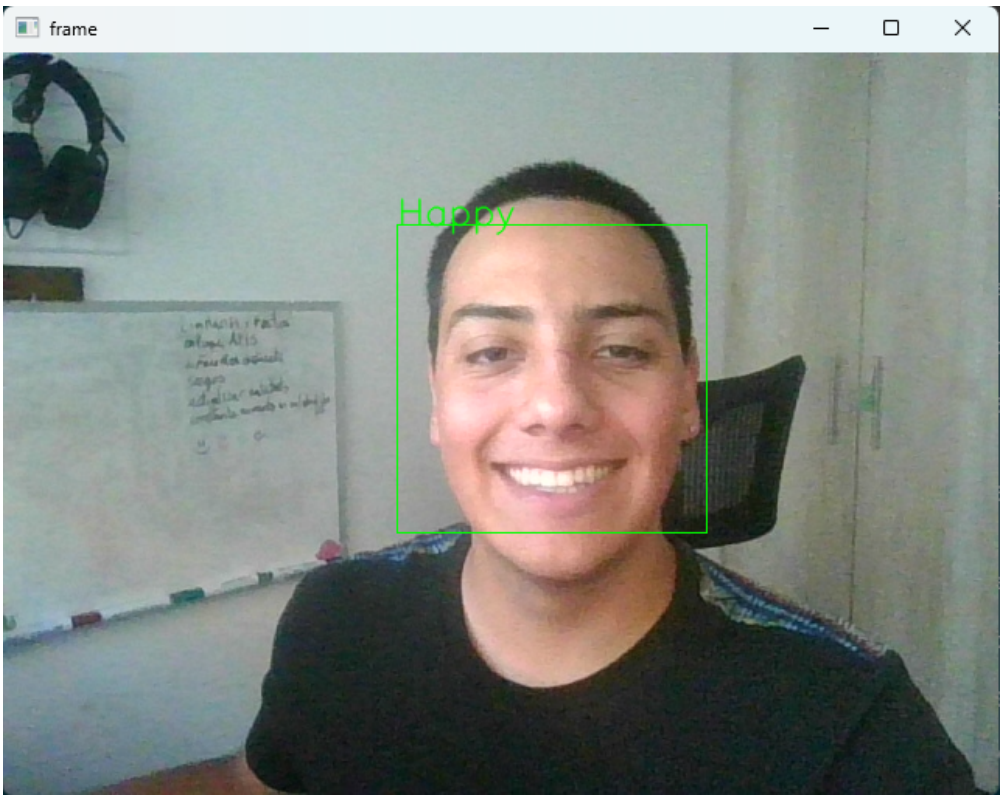
Procedemos a realizar el entrenamiento del modelo

```
# Entrenar el modelo
model.fit(X_train, y_train, batch_size=32, epochs=30, validation_data=(X_val, y_val))
✓ 1m 45.1s
Epoch 1/30
16/16 [==============================] - 21s 235ms/step - loss: 122.5404 - accuracy: 0.3148
Epoch 2/30
16/16 [==============================] - 3s 198ms/step - loss: 0.7626 - accuracy: 0.7398 -
Epoch 3/30
16/16 [==============================] - 3s 198ms/step - loss: 0.2909 - accuracy: 0.9016 -
Epoch 4/30
16/16 [==============================] - 3s 196ms/step - loss: 0.1095 - accuracy: 0.9531 -
Epoch 5/30
16/16 [==============================] - 3s 181ms/step - loss: 0.0866 - accuracy: 0.9602 -
Epoch 6/30
16/16 [==============================] - 3s 176ms/step - loss: 0.0464 - accuracy: 0.9762 -
Epoch 7/30
```

Reconocimiento de expresiones faciales







En esta ultima fase se utiliza cv2 para realizar captura y validacion del rostro en tiempo real por medio de la webcam

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

BASEPATH = './'
MODELPATH = './diego.h5'

#Prueba en tiempo real del modelo del modelo

emotion_dict = {0: "Angry", 1: "Happy", 2: "Sad", 3: "Surprise",
4: "Neutral"}

model = load_model('./diego.h5')

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
```

```
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0),
1)
        roi_gray = gray[y:y + h, x:x + w]
        cropped_img =
np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)),
-1), 0)
        cv2.normalize(cropped_img, cropped_img, alpha=0, beta=1,
norm_type=cv2.NORM_L2, dtype=cv2.CV_32F)
        prediction = model.predict(cropped_img)
        cv2.putText(frame,
emotion_dict[int(np.argmax(prediction))],
        (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1,
cv2.LINE_AA)

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

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

Conclusiones

Con este proyecto pudimos identificar el uso de una CNN en tiempo real, y con esto, ver el potencial que estas técnicas tienen diferentes medios como cámaras de celular, fotográficas, celulares, sistemas de seguridad entre otros, ya que es claro que todos estos pueden ser potenciados en tiempo real por algún modelo específico.

Referencias

- FER-2013. (2020, 19 julio). Kaggle. FER-2013 | Kaggle
- Sharma, H. (2021). Create Your own Image Dataset using Opencv in Machine Learning. Analytics Vidhya. Create Your own Image Dataset using Opencv in Machine Learning - Analytics Vidhya
- OpenCV. (2023, 9 mayo). Home - OpenCV. OpenCV
- Thetechwriters. (2022). Facial Emotion Detection Using CNN. Analytics Vidhya. Facial Emotion Detection Using CNN - Analytics Vidhya