TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CULIACÁN

CARRERA

ING. EN SISTEMAS COMPUTACIONALES

MATERIA

INTELIGENCIA ARTIFICIAL

UNIDAD 4

TAREA 1

ALUMNOS

TRUJILLO ACOSTA BRYANT

BRAYAN MENDOZA GARCIA

PROFESOR

MORA FÉLIX ZURIEL DATHAN

Encontramos este dataset cuenta con 7 emociones distinas que son enojado, disgustado, miedo, feliz, neutral, triste y sorprendido, tiene una gran cantidad de imágenes con diferentes emociones y este las clasifica , lo verificamos para ver que todas estén correctas y funciona bien, no importa la edad, ni tono de piel ni nada, solo se basa en la emoción.

https://www.kaggle.com/datasets/msambare/fer2013/data

El primer paso vamos a utilizar las siguientes librerías para poder preprocesar las imágenes

```
import cv2
import os
import numpy as np
from tqdm import tqdm
```

- **cv2:** Librería OpenCV para procesamiento de imágenes.
- **os:** Permite trabajar con rutas de carpetas y archivos.
- **numpy:** Para manipular matrices e imágenes.
- **tqdm:** Muestra una barra de progreso en consola mientras se procesan las imágenes.

Definir los siguientes parámetros

```
# Parámetros de configuracion
IMG_SIZE = 48
input_folder = 'train'
output_folder = 'augmented'
```

- **IMG_SIZE:** Tamaño al que se redimensionarán las imágenes (48x48 píxeles).
- **input_folder:** Carpeta donde están las imágenes originales organizadas por clases (emociones).
- **output_folder:** Carpeta donde se guardarán las imágenes aumentadas.

función para ajustar el Brillo y el contraste

```python
# Aumentar o disminuir brillo
def adjust_brightness(img, value):
    hsv = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
    hsv = cv2.cvtColor(hsv, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v, value)
    final_hsv = cv2.merge((h, s, v))
    img_bright = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
    return cv2.cvtColor(img_bright, cv2.COLOR_BGR2GRAY)
```

Cambia el brillo de la imagen modificando el canal V del espacio HSV.

función Para poder rotar la imagen

```python
# Rotar imagen
def rotate_image(img, angle):
    center = (IMG_SIZE // 2, IMG_SIZE // 2)              # Centro de rotación
    M = cv2.getRotationMatrix2D(center, angle, 1.0)      # Matriz de rotación
    return cv2.warpAffine(img, M, (IMG_SIZE, IMG_SIZE))  # Aplicar rotación
```

Rota la imagen usando una matriz de transformación respecto al centro.

Función para escalar (hacer zoom)

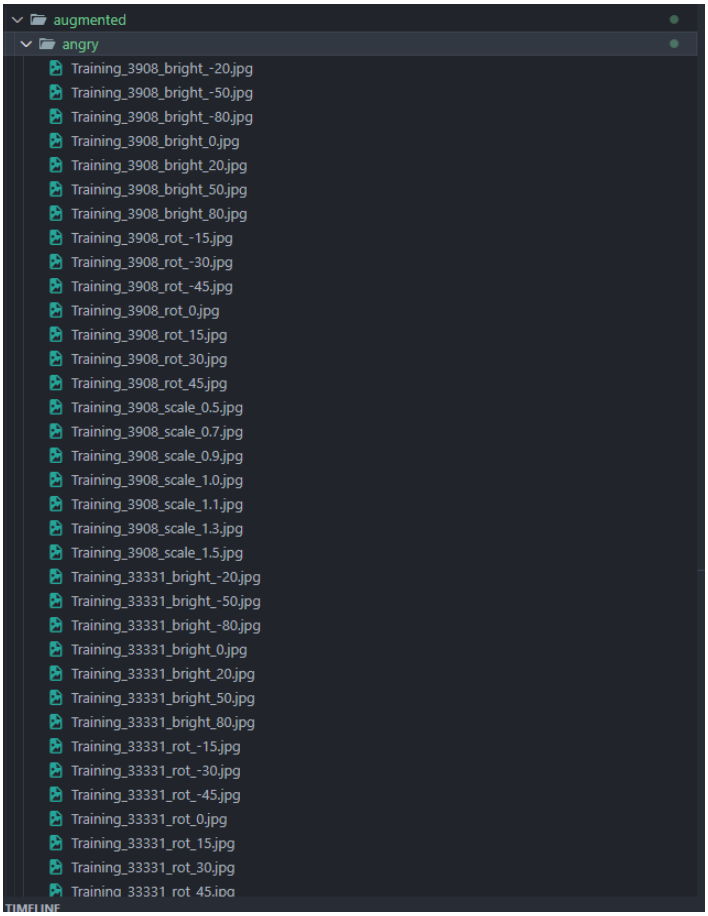
```python
# Escalar imagen (zoom)
def scale_image(img, scale):
    h, w = img.shape[:2]
    new_h, new_w = int(h * scale), int(w * scale)
    resized = cv2.resize(img, (new_w, new_h))

    if scale > 1.0:
        # Recortar centro si imagen es más grande
        start_h = (new_h - h) // 2
        start_w = (new_w - w) // 2
        return resized[start_h:start_h+h, start_w:start_w+w]
    else:
        # Poner imagen pequeña al centro
        output = np.zeros((h, w), dtype=np.uint8)
        start_h = (h - new_h) // 2
        start_w = (w - new_w) // 2
        output[start_h:start_h+new_h, start_w:start_w+new_w] = resized
        return output
```

Calcula el nuevo tamaño de la imagen según la escala dada, Si scale > 1.0: recorta el centro, Si scale < 1.0: coloca la imagen pequeña en el centro con padding

Función para aplicar aumentaciones y guardar

```python
# Aumentación
def augment_and_save(img, label, filename):
    base_path = os.path.join(output_folder, label)
    os.makedirs(base_path, exist_ok=True)
    name = os.path.splitext(filename)[0]
```

Crea una carpeta con el nombre de la clase (ej. feliz, triste, etc). Separa el nombre del archivo para usarlo como base para los nombres nuevos.



Este es un ejemplo de como quedaría guardado las imágenes preprocesadas, es una captura de la carpeta que fue creada al preprocesar las imágenes, no se puede subir a github ya que sobrepasa el tamaño como para subirlo

```python
# Brillo
brightness_levels = [-80, -50, -20, 0, 20, 50, 80]
for i, b in enumerate(brightness_levels):
    bright_img = adjust_brightness(img, b) if b != 0 else img
    cv2.imwrite(os.path.join(base_path, f'{name}_bright_{b}.jpg'), bright_img)
```

Se crean 7 imágenes con distintos niveles de brillo.

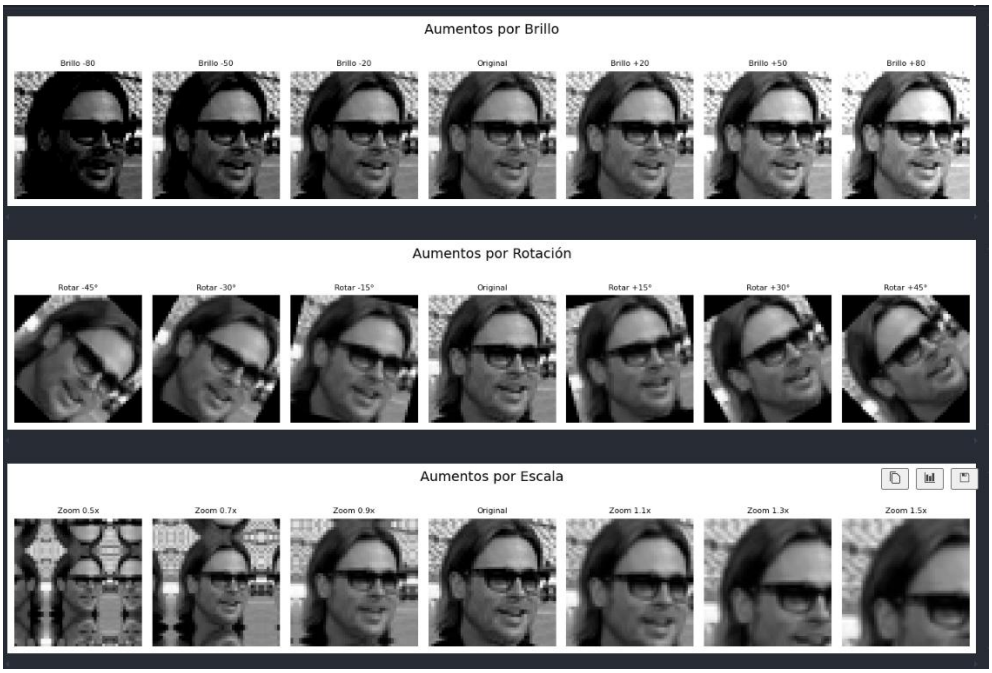
```python
# Rotaciones
angles = [-45, -30, -15, 0, 15, 30, 45]
for a in angles:
    rot_img = rotate_image(img, a) if a != 0 else img
    cv2.imwrite(os.path.join(base_path, f'{name}_rot_{a}.jpg'), rot_img)
```

Se crean 7 imágenes rotadas desde -45° hasta +45°.

```python
# Escalado
scales = [0.5, 0.7, 0.9, 1.0, 1.1, 1.3, 1.5]
for s in scales:
    scaled_img = scale_image(img, s) if s != 1.0 else img
    cv2.imwrite(os.path.join(base_path, f'{name}_scale_{s:.1f}.jpg'), scaled_img)
```

Se crean 7 imágenes con diferentes niveles de zoom.

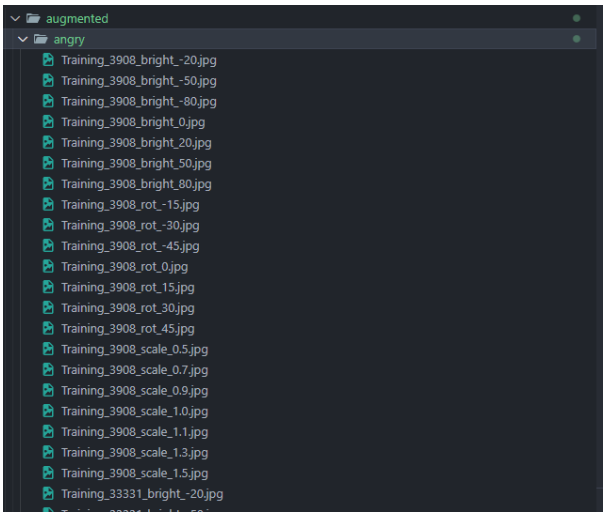
RESULTADOS



Bueno el resultado que haría en cada una de las imágenes seria este

```
rs/payin/Inteligencia-Artificial/UNIDAD4/Tarea_1/app.py
Procesando angry: 100%|| 3995/3995 [00:43<00:00, 92.40it/s]
Procesando disgust: 100%|| 436/436 [00:04<00:00, 95.83it/s]
Procesando fear: 100%|| 4097/4097 [00:45<00:00, 90.50it/s]
Procesando happy: 100%|| 7215/7215 [01:31<00:00, 78.97it/s]
Procesando neutral: 100%|| 4965/4965 [00:55<00:00, 89.87it/s]
Procesando sad: 100%|| 4830/4830 [00:50<00:00, 96.43it/s]
Procesando surprise: 100%|| 3171/3171 [00:30<00:00, 102.65it/s]
PS C:\Users\payin\Inteligencia-Artificial\UNIDAD4\Tarea_1>
```

Esto fue las imágenes que fueron pre procesadas



Esto es como se guardaron las imágenes con cada una de las aumentaciones que se hicieron de brillo, escala, rotacion.

FUENTES

EL DATA SET FUE TOMADO DE
https://www.kaggle.com/datasets/msambare/fer2013?resource=download
El procesamiento fue gracias al siguiente ejemplo que tomamos de kaggle, nos facilito el entendimiento para hacerlo

https://www.kaggle.com/code/hassanmuhammed1/facial-emotion-detection/notebook