



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de México

Campus Culiacán

Ingeniería en Sistemas Computacionales

Inteligencia Artificial

Detector de emociones

Docente:

-Zuriel Dathan Mora Felix

Equipo:

-Yosef Emiliano Soto Iribé

-Cristian Andrea Ramírez Medina

Arquitectura neuronal

Para este proyecto se optó por implementar una **red neuronal convolucional**, esto debido a que es una arquitectura que se adapta perfectamente al problema que se nos está presentando, en este caso lo que nosotros buscamos es que nuestra red neuronal aprenda de un dataset compuesto por puras imágenes, y para ello este tipo de red neuronal es perfecta.

En la primera capa convolucional de mi red neuronal se aplica una capa con 32 filtros con tamaño de 3x3 con la función de activación RELU, esta capa está configurada para recibir imágenes de tamaño 100x100 que es el tamaño por el que por defecto vienen las imágenes en el dataset es por ello que se configuró para eso, después de ello se aplica una capa de max pooling con una ventana de 2x2 la cual reduce la dimensión espacial y ayuda a extraer características importantes como bordes y contornos.

Para la segunda capa se aumentó la cantidad de filtros a 64 y se dejó el tamaño de 3x3 así como también la misma función de activación RELU.

La tercera capa continua profundizando el aprendizaje de características usando 128 filtros y también con la función de activación RELU.

También se implementó una capa de aplanamiento (Flatten) en la que la salida de tres dimensiones de las capas convolucionales se transforma en un vector de una dimensión, eso es necesario para poder conectar la representación extraída con las capas densas posteriores, también se incluye una capa densa (Dense) esta capa se conecta con 256 neuronas y usamos también la función de activación RELU, la función de esta capa es actuar como una red neuronal tradicional combinando características extraídas para tomar decisiones.

También se implementó una capa de regularización (Dropout) en la que usa la técnica de regularización llamada dropout la cual desactiva de manera aleatoria la mitad de las neuronas cuando se está haciendo el entrenamiento, esto lo que hace es ayudar a prevenir el sobreajuste y hace que el modelo no dependa de mas con neuronas específicas.

Al último se implementa la capa de salida (Dense) con un número de neuronas igual al número de clases de emociones, esta capa usa la función de activación softmax en la que asigna una probabilidad de 0 a 1 cada clase.

En todas y cada una de las capas ocultas se implementa la función de activación RELU, esta para introducir no linealidad al modelo y que pueda aprender relaciones complejas entre los datos, para la capa de salida se implementó la función softmax esto porque transforma la salida en una distribución de probabilidad.

Código:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size,
img_size, 1)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
```

```
        MaxPooling2D(2, 2),  
  
        Conv2D(128, (3, 3), activation='relu'),  
        MaxPooling2D(2, 2),  
  
        Flatten(),  
        Dense(256, activation='relu'),  
        Dropout(0.5),  
  
        Dense(train_generator.num_classes, activation='softmax')  
    )
```

Definición de parámetros

En la definición de parámetros definimos aquellos que serán usados para el entrenamiento del modelo, mi caso definimos los siguientes:

```
img_size = 100  
batch_size = 32  
epochs = 15  
train_dir = 'DATASET/train'  
test_dir = 'DATASET/test'
```

El tamaño de la imagen es donde defino el tamaño al que se redimensiona todas las imágenes antes de ser ingresadas al modelo, en nuestro caso la imagen será de 100x100 píxeles.

El batch size especifica cuántas imágenes se procesan de forma simultánea durante una iteración del entrenamiento, en mi caso definimos 32.

En la parte de epochs le indicamos cuantas veces verá el modelo todo el conjunto de entrenamiento completo, algo que hay que aclarar es que mientras más epochs permitamos hace que el modelo aprenda mejor sin embargo también aumenta el riesgo de que el modelo empiece a sobre ajustarse o a memorizar de alguna manera el conjunto de entrenamiento, nosotros definimos 15 porque consideramos que es un número moderado que nos puede ser útil para observar cómo evoluciona la precisión sin entrenar el modelo de más.

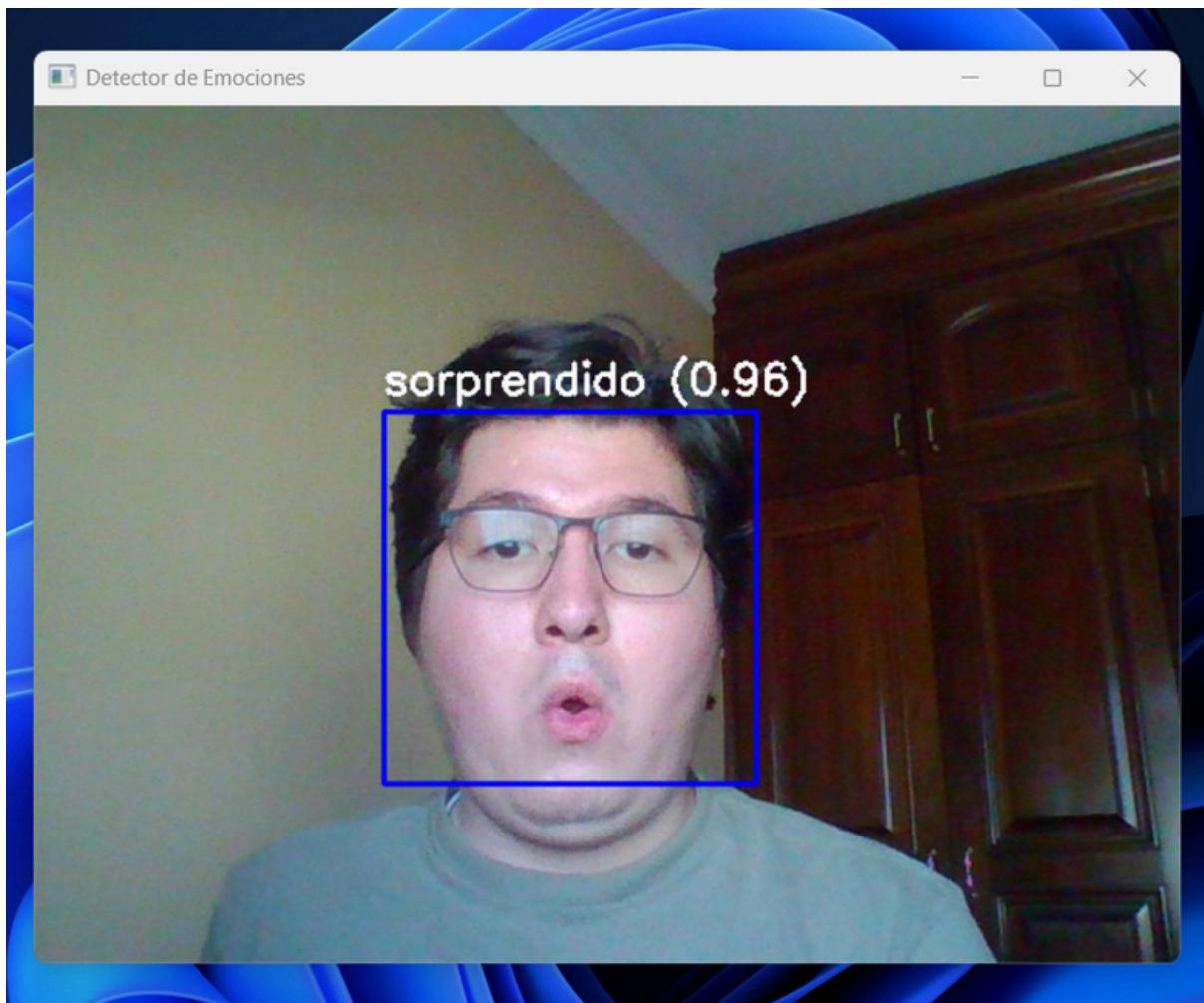
```
PROBLEMS 6 OUTPUT TERMINAL PORTS DEBUG CONSOLE

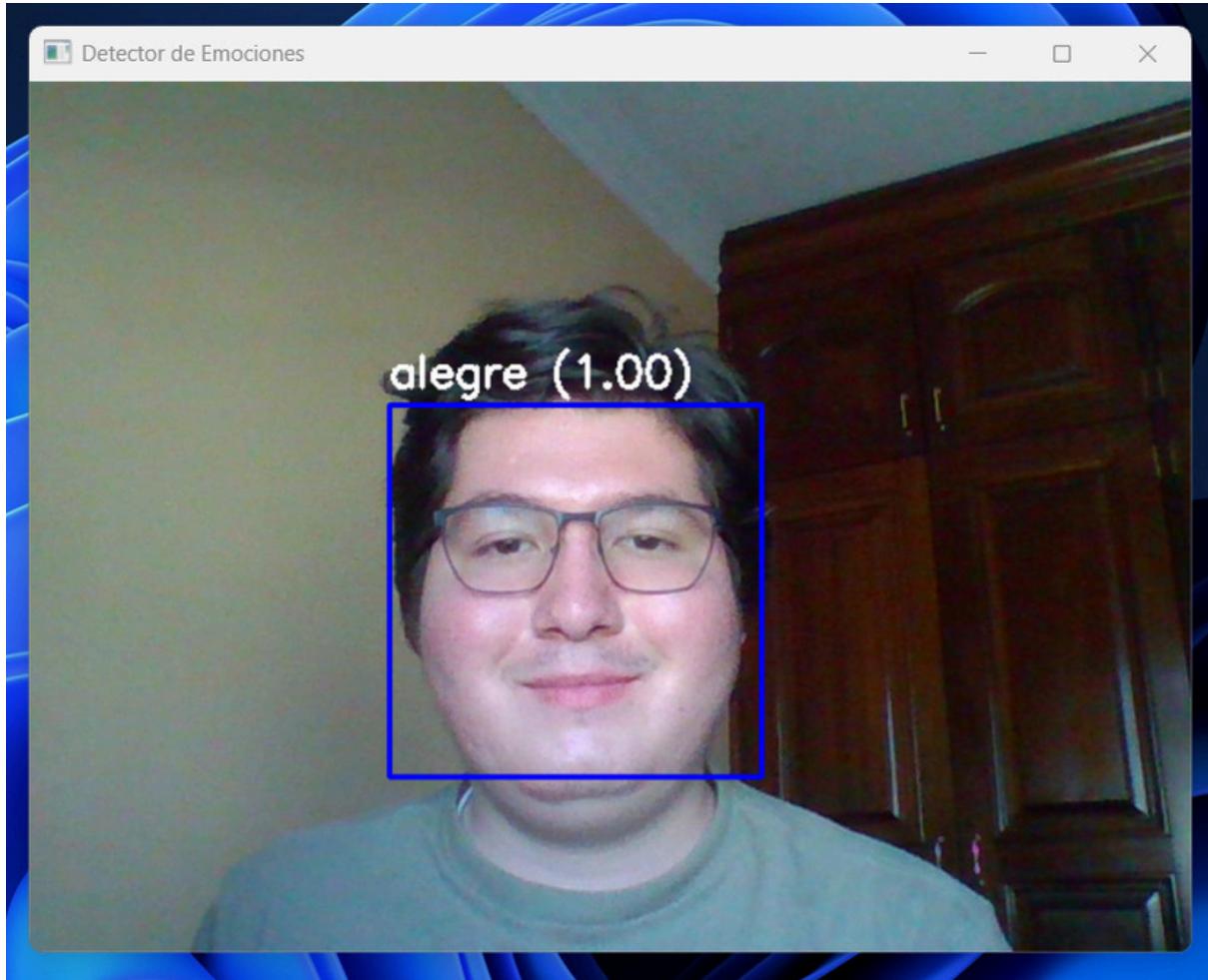
self._warn_if_super_not_called()
Epoch 1/15
283/283 27s 92ms/step - accuracy: 0.5682 - loss: 1.1497 - val_accuracy: 0.7522 - val_loss: 0.7278
Epoch 2/15
283/283 26s 90ms/step - accuracy: 0.7455 - loss: 0.7175 - val_accuracy: 0.7908 - val_loss: 0.6022
Epoch 3/15
283/283 26s 91ms/step - accuracy: 0.7998 - loss: 0.5811 - val_accuracy: 0.8294 - val_loss: 0.4986
Epoch 4/15
283/283 26s 92ms/step - accuracy: 0.8330 - loss: 0.4790 - val_accuracy: 0.8461 - val_loss: 0.4595
Epoch 5/15
283/283 26s 92ms/step - accuracy: 0.8660 - loss: 0.3987 - val_accuracy: 0.8452 - val_loss: 0.4378
Epoch 6/15
283/283 26s 92ms/step - accuracy: 0.8763 - loss: 0.3491 - val_accuracy: 0.8523 - val_loss: 0.4260
Epoch 7/15
283/283 26s 91ms/step - accuracy: 0.8987 - loss: 0.3011 - val_accuracy: 0.8474 - val_loss: 0.4728
Epoch 8/15
283/283 26s 92ms/step - accuracy: 0.9102 - loss: 0.2478 - val_accuracy: 0.8559 - val_loss: 0.4639
Epoch 9/15
283/283 26s 92ms/step - accuracy: 0.9232 - loss: 0.2222 - val_accuracy: 0.8443 - val_loss: 0.4629
Epoch 10/15
283/283 26s 92ms/step - accuracy: 0.9326 - loss: 0.1901 - val_accuracy: 0.8532 - val_loss: 0.4406
Epoch 11/15
283/283 26s 92ms/step - accuracy: 0.9469 - loss: 0.1567 - val_accuracy: 0.8604 - val_loss: 0.5098
Epoch 12/15
283/283 26s 92ms/step - accuracy: 0.9525 - loss: 0.1359 - val_accuracy: 0.8595 - val_loss: 0.5656
Epoch 13/15
283/283 26s 92ms/step - accuracy: 0.9616 - loss: 0.1115 - val_accuracy: 0.8689 - val_loss: 0.5700
Epoch 14/15
283/283 26s 91ms/step - accuracy: 0.9619 - loss: 0.1040 - val_accuracy: 0.8671 - val_loss: 0.5502
Epoch 15/15
283/283 26s 92ms/step - accuracy: 0.9692 - loss: 0.0856 - val_accuracy: 0.8636 - val_loss: 0.6132
WARNING:abs1:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered
1.keras)` or `keras.saving.save_model(model, 'my_model.keras')`.
(.venv) PS D:\detector_emociones>
```

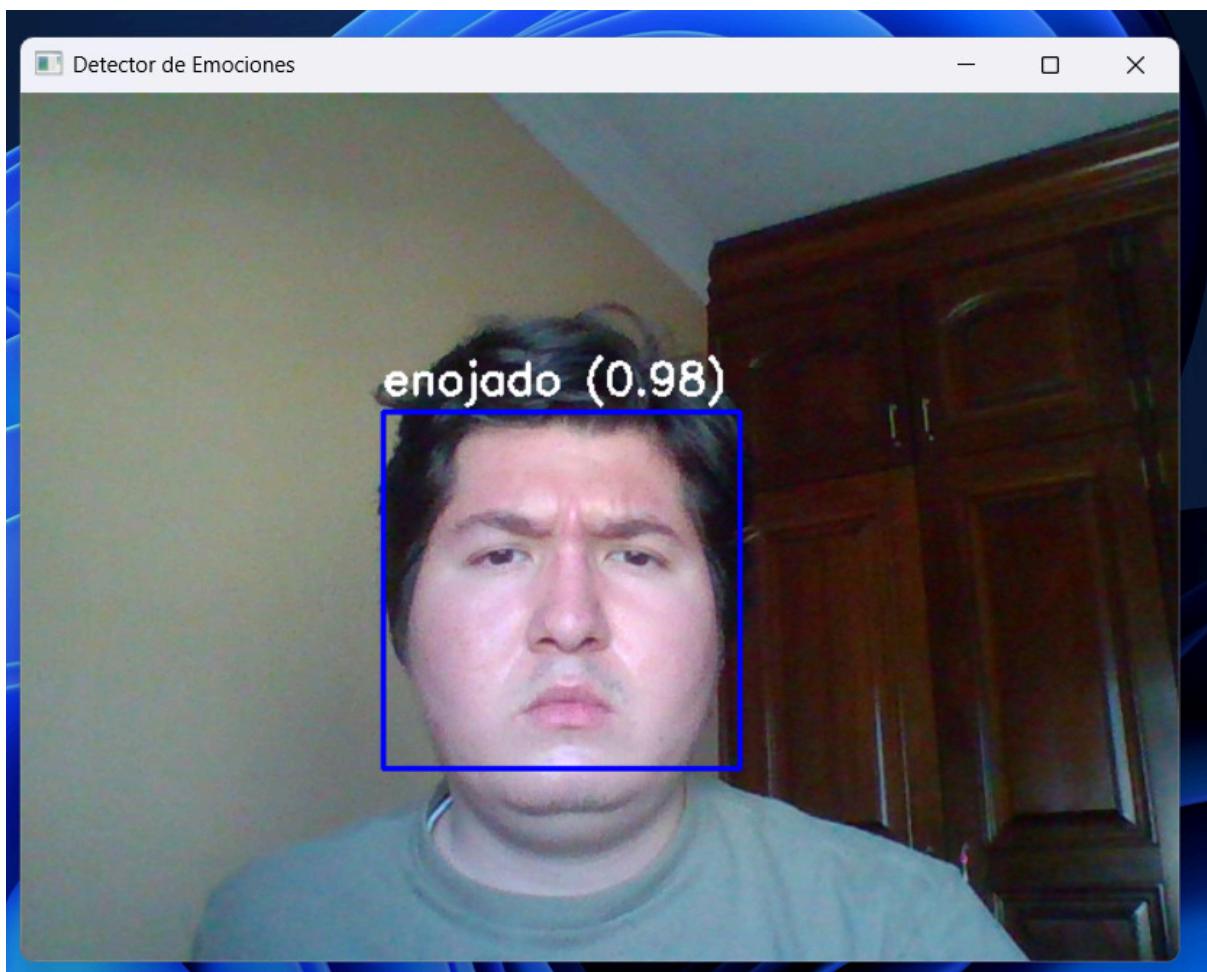
Los otros dos parámetros que definimos son los directorios en los que están los conjuntos de entrenamiento y de prueba.

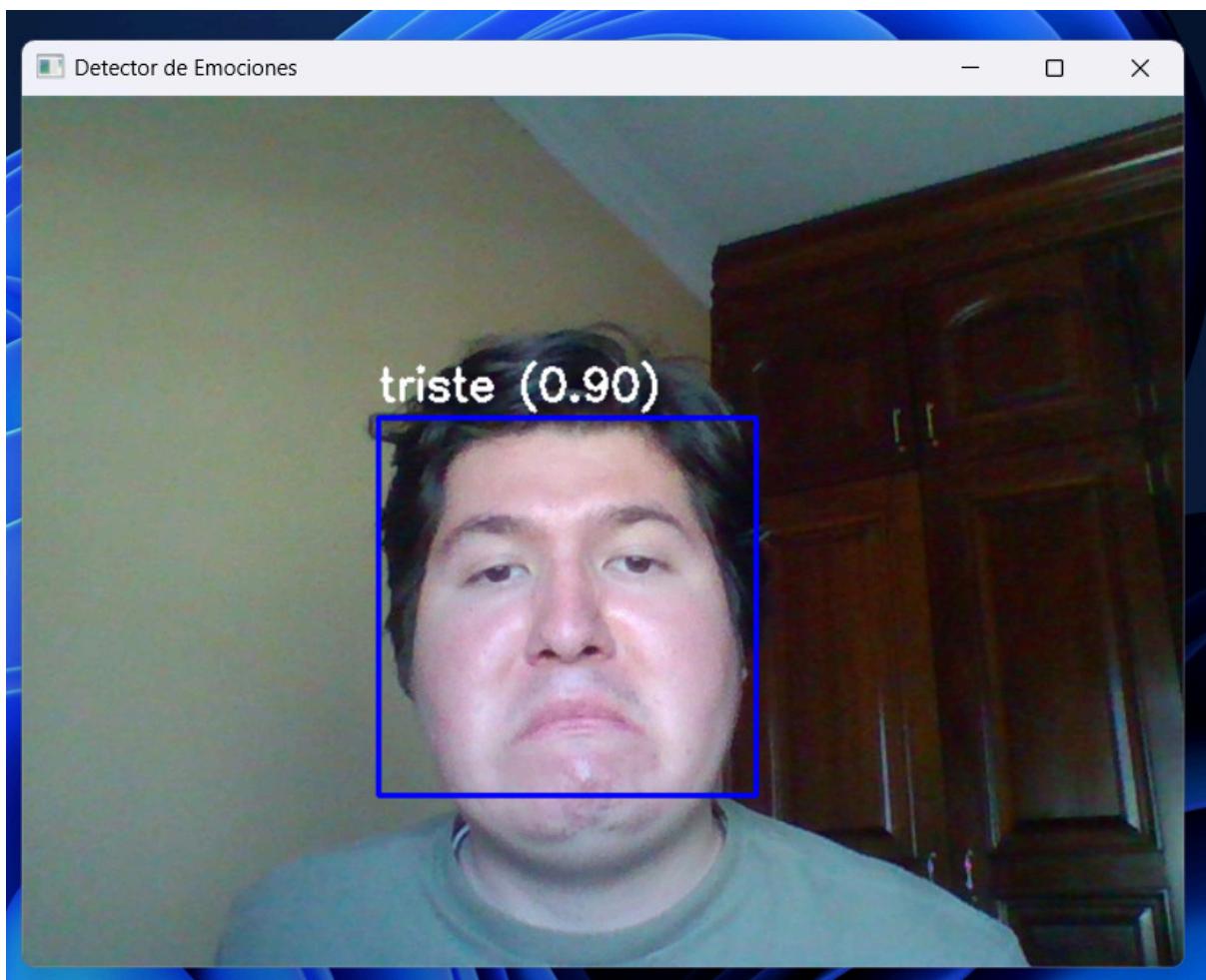
Pruebas

En las siguientes imágenes podemos ver el programa en ejecución en el cual utilizando la webcam este detecta algunas de las emociones soportadas









Validación del modelo

Una parte importante en el desarrollo de todo modelo es precisamente su validación, esto lo logramos usando una matriz de confusión la cual es una tabla que visualiza las predicciones de un modelo de clasificación, mostrando cuántas instancias de cada clase fueron correctamente clasificadas y cuántas fueron confundidas con otras clases, su uso es importante porque ofrece una validación más completa y detallada, aquí podemos darnos cuenta como en su mayoría la mayor parte de los casos de prueba han sido acertados para cada una de las emociones soportadas por el modelo.



Reporte de Clasificación:				
	precision	recall	f1-score	support
alegre	0.90	0.95	0.92	1185
angustiado	0.67	0.41	0.50	74
enojado	0.77	0.67	0.71	162
sorprendido	0.84	0.79	0.82	329
triste	0.81	0.81	0.81	478
accuracy			0.86	2228
macro avg	0.80	0.72	0.75	2228
weighted avg	0.85	0.86	0.85	2228

Documentación del código

Lo primero que hacemos en el programa es precisamente importar las librerías que vamos a utilizar, como por ejemplo la librería os que nos sirve para la manipulación de sistemas de archivos, numpy para operaciones numéricas y manejo de arreglos, bibliotecas para visualizar lo que es la matriz de confusión como matplotlib, tensorflow para todo el tema de las redes neuronales y scikit-learn para que nos permita medir el rendimiento del modelo con metrics.

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

Luego lo que hacemos es definir varios parámetros importantes para entrenar el modelo como el tamaño de las imágenes, el tamaño por lote, y las epochs que se realizarán para el entrenamiento, así como también los directorios donde se encuentran las imágenes para el entrenamiento.

```
img_size = 100
batch_size = 32
epochs = 15
train_dir = 'DATASET/train'
test_dir = 'DATASET/test'
```

Luego preparamos los generadores de datos con el ImageDataGenerator donde creamos dos generadores uno para el entrenamiento y el otro para el test, estos también normalizan los valores de los pixeles dividiendo por 255 para transformar los valores originales de 0 a 255 en un rango de 0 a 1.

```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical'
)
```

```
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical',
    shuffle=False
)
```

Después se empieza a definir lo que es la red neuronal convolucional utilizando tensorflow y keras.

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size,
img_size, 1)),
    MaxPooling2D(2, 2),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),

    Dense(train_generator.num_classes, activation='softmax')
])
```

El modelo se compila usando el optimizador de adams y se mide la precisión accuracy como métrica principal, después el modelo es entrenado utilizando el generador de datos de entrenamiento, para después guardarse en un archivo .h5

```
model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_generator, epochs=epochs,
validation_data=test_generator)

model.save("modelo_emociones_nuevo.h5")
```

Código completo modelo.py

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam

img_size = 100
batch_size = 32
epochs = 15
train_dir = 'DATASET/train'
test_dir = 'DATASET/test'

train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical',
    shuffle=False
)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size,
img_size, 1)),
```

```

        MaxPooling2D(2, 2),

        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),

        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),

        Flatten(),
        Dense(256, activation='relu'),
        Dropout(0.5),

        Dense(train_generator.num_classes, activation='softmax')
    )

model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_generator, epochs=epochs,
validation_data=test_generator)

model.save("modelo_emociones_nuevo.h5")

#Matriz de confusión
Y_pred = model.predict(test_generator, steps=test_generator.samples // 
test_generator.batch_size + 1)
y_pred = np.argmax(Y_pred, axis=1)

y_true = test_generator.classes

target_names = list(test_generator.class_indices.keys())

cm = confusion_matrix(y_true, y_pred)
print("Reporte de Clasificación:")
print(classification_report(y_true, y_pred, target_names=target_names))

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=target_names, yticklabels=target_names)
plt.xlabel('Predicho')
plt.ylabel('Real')
plt.title('Matriz de Confusión')
plt.tight_layout()
plt.show()

```

app.py

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import img_to_array

model = load_model("modelo_emociones_nuevo.h5")
labels = ['alegre', 'angustiado', 'enojado', 'sorprendido', 'triste']

face_classifier = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
img_size = 100
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5)

    for (x, y, w, h) in faces:
        roi = gray[y:y+h, x:x+w]
        roi = cv2.resize(roi, (img_size, img_size))
        roi = roi.astype("float") / 255.0
        roi = img_to_array(roi)
        roi = np.expand_dims(roi, axis=0)

        pred = model.predict(roi)[0]
        label = labels[np.argmax(pred)]
        confianza = np.max(pred)

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        cv2.putText(frame, f"{label} ({confianza:.2f})", (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)

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

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

Link de video de demostración

<https://www.youtube.com/watch?v=va8aYJfSAWQ>