UNIVERSIDAD NACIONAL DE SAN CRISTOBAL DE HUAMANGA

FACULTAD DE INGENIERIA DE MINAS, GEOLOGIA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

MACHINE LEARNING PARA LA DETECCION DETECCIONES DE EMOCIONES EN TIEMPO REAL

CURSO : LAB Sistemas Expertos (IS-442)

PROFESOR : HUARANCCA ÑAUPARI, YONNY

INTEGRANTES : INFANZON GOMEZ, Yon Roussvett – 27190114

AYACUCHO-PERÚ

2023

TABLA DE CONTENIDO

RESUMEN

Este proyecto se centra en desarrollar un sistema de reconocimiento de emociones en tiempo real utilizando redes neuronales. En la actualidad, la inteligencia artificial y el aprendizaje automático están transformando diferentes campos tecnológicos, y la capacidad de las máquinas para comprender las emociones humanas es un aspecto crucial. El objetivo principal es crear un sistema que interprete expresiones faciales en tiempo real y reconozca emociones como alegría, tristeza, enfado, sorpresa y miedo. El uso de redes neuronales permitirá que la máquina aprenda a identificar patrones en las expresiones faciales y clasificar las emociones. Este enfoque tiene aplicaciones en marketing, publicidad, experiencia de usuario y terapias emocionales.

ABSTRACT

In today's context, artificial intelligence and machine learning are revolutionizing various technological fields, and a crucial aspect is the ability to recognize and understand human emotions. This project focuses precisely on developing a real-time emotion recognition system using neural networks. Emotions play a pivotal role in our daily lives, influencing decisions, perceptions, and behaviors. While emotion detection is a natural human skill, equipping machines with this capability has been a significant and valuable challenge in artificial intelligence research. The primary goal of this project is to create a system capable of capturing and analyzing real-time facial expressions, enabling machines to interpret emotions such as joy, sadness, anger, surprise, and fear. This opens up possibilities in diverse areas, from marketing and advertising to user experience assessment and emotional therapy.

CAPITULO 1. INTRODUCCIÓN

En la actualidad, la inteligencia artificial y el aprendizaje automático están revolucionando diversos campos de la tecnología, y uno de los aspectos más importantes es la capacidad de reconocer y comprender las emociones humanas. Este proyecto se enfoca en desarrollar precisamente eso: un sistema de reconocimiento de emociones en tiempo real utilizando redes neuronales.

Las emociones juegan un papel crucial en nuestra vida cotidiana, afectando nuestras decisiones, percepciones y comportamientos. La detección de emociones es una habilidad humana natural, pero dotar a las máquinas con esta capacidad ha sido un desafío importante y valioso para la investigación en inteligencia artificial.

El objetivo principal de este proyecto es desarrollar un sistema que pueda capturar y analizar nuestras expresiones faciales en tiempo real, permitiendo que las máquinas interpreten nuestras emociones, como alegría, tristeza, enfado, sorpresa y miedo, entre otras. Esto abre un mundo de posibilidades en diversas áreas, desde el marketing y la publicidad hasta la evaluación de experiencias de usuario y la asistencia en terapias emocionales.

Para lograr este propósito, utilizaremos redes neuronales, que son un tipo de modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Estas redes son capaces de aprender patrones complejos y realizar tareas de clasificación y reconocimiento con precisión. Las redes neuronales nos permitirán enseñar a la máquina a identificar y clasificar diferentes emociones basándose en las características y patrones presentes en las expresiones faciales.

1.1 Antecedenes

El reconocimiento de emociones a través de expresiones faciales ha sido objeto de investigación durante muchos años. Los avances en el campo del aprendizaje automático, en particular en las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN), han mejorado significativamente la precisión y eficiencia de estos sistemas.(Avilés Valencia et al., 2022)

1.2 Objetivos

Desarrollar un sistema de reconocimiento de emociones en tiempo real utilizando redes neuronales para analizar las expresiones faciales capturadas por una cámara.
Objetivos específicos:

- Recopilar un conjunto de datos etiquetados que contenga imágenes faciales con expresiones emocionales variadas.
- Preprocesar y normalizar las imágenes faciales para su uso en el entrenamiento de redes neuronales.
- Entrenar una red neuronal convolucional o una red neuronal recurrente utilizando los datos etiquetados para que pueda reconocer y clasificar las emociones presentes en las expresiones faciales.
- Implementar un sistema que capture las expresiones faciales en tiempo real a través de una cámara y utilice el modelo entrenado para reconocer y clasificar las emociones.
- Evaluar la precisión y eficacia del sistema utilizando métricas de desempeño y pruebas con usuarios reales.
- Realizar ajustes y mejoras en el sistema en función de los resultados de la evaluación.

1.3 Justificación

- Relevancia Social y Psicológica: El reconocimiento de emociones en tiempo real tiene una gran relevancia social y psicológica. Ayudará a comprender y mejorar las interacciones humanas con la tecnología, lo que puede tener un impacto positivo en la calidad de vida de las personas al brindar respuestas más personalizadas y empáticas.(Cordero & Aguilar, 2016)

- Potencial en el Mercado: La aplicación de esta tecnología en el marketing y la publicidad podría proporcionar una valiosa información sobre cómo los productos y campañas afectan emocionalmente a los clientes, lo que permitiría ajustar las estrategias de manera más efectiva.(Ribes Gil, 2017)

1.4 Delimitación

- Alcance del Reconocimiento Emocional: Este proyecto se enfocará en el reconocimiento de seis emociones básicas: alegría, tristeza, enfado, sorpresa, miedo y neutralidad. Si bien existen muchas otras emociones complejas y sutiles, abordar estas seis emociones es un buen punto de partida para demostrar la viabilidad del sistema en tiempo real.

- Tipo de Datos: El sistema se centrará en el análisis de expresiones faciales capturadas por una cámara. No abarcará otras fuentes de datos, como el reconocimiento de emociones a partir de la voz o el lenguaje escrito.

- Dispositivos de Captura: El sistema estará diseñado para funcionar en computadoras con cámaras integradas o cámaras externas conectadas. No abarcará dispositivos móviles o sistemas embebidos, aunque la arquitectura se pueda adaptar para su uso en diferentes plataformas en futuros desarrollos.

CAPITULO 2. MARCO TEORICO

2.1 Librerías

2.1.1 Numpy

NumPy constituye el pilar esencial de la computación científica en el entorno de programación Python. Se trata de una librería que ofrece un objeto de matriz de múltiples dimensiones, así como otros objetos derivados como matrices y matrices enmascaradas. Asimismo, dispone de una amplia gama de funciones que posibilitan operaciones de alta velocidad en matrices, abarcando tareas de manipulación matemática, operaciones lógicas, ajuste de dimensiones, ordenamiento, selección, entrada/salida de datos, transformadas de Fourier discretas, conceptos básicos de álgebra lineal, funciones estadísticas elementales, generación de simulaciones aleatorias y una diversidad de funcionalidades adicionales.

2.1.2 Pandas

Pandas representa una de las herramientas más valiosas para los científicos de datos que trabajan con Python. En este conjunto de bibliotecas, las entidades de datos primordiales se identifican como Series, empleadas para datos unidimensionales, y DataFrame, diseñadas para datos bidimensionales. Dichas estructuras son ampliamente empleadas en diversos dominios, que incluyen finanzas, estadísticas, ciencias sociales y múltiples áreas de la ingeniería. Pandas se distingue por su enfoque intuitivo y flexible, que simplifica tanto la manipulación como el análisis de datos de manera significativa.

2.1.3 Matplotlib

Matplotlib es la principal y más reconocida biblioteca gráfica de Python. Te permite producir representaciones visuales de alta calidad que son aptas tanto para publicaciones impresas como para formatos digitales. Con Matplotlib, tienes la capacidad de generar una amplia variedad de gráficos, abarcando desde series temporales, histogramas y espectros de potencia, hasta gráficos de barras, gráficos de errores y otros tipos de representaciones.

2.2 Aprendizaje supervisado

El aprendizaje supervisado es un enfoque en el aprendizaje automático donde un algoritmo o modelo se entrena utilizando un conjunto de datos que contiene ejemplos etiquetados. En este contexto, "etiquetados" significa que cada ejemplo del conjunto de datos está asociado con una etiqueta o resultado conocido. El objetivo del aprendizaje supervisado es que el modelo aprenda a hacer predicciones precisas en nuevos datos no

etiquetados, basándose en la relación entre las características de entrada y las etiquetas conocidas del conjunto de entrenamiento.

2.3 Red Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN) son un tipo de arquitectura de red neural especializada en el procesamiento de datos con estructura espacial, como imágenes. Están diseñadas para extraer automáticamente características relevantes de las imágenes mediante el uso de operaciones de convolución y pooling, lo que las hace especialmente adecuadas para el reconocimiento de patrones en expresiones faciales.

2.4 Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes (RNN) son otro tipo de arquitectura de red neural que se utiliza para analizar datos secuenciales, como el lenguaje natural o señales de tiempo continuo. Su diseño les permite mantener una memoria interna que les permite trabajar con datos en secuencia y capturar dependencias temporales, lo que las hace útiles para el reconocimiento de patrones en secuencias de imágenes o señales, como la modulación de las expresiones faciales a lo largo del tiempo.

2.5 Conjunto de datos y preprocesamiento

La calidad y diversidad del conjunto de datos son fundamentales para el entrenamiento de una red neuronal eficaz. Se abordará la importancia de la recopilación y etiquetado de un conjunto de datos equilibrado que contenga expresiones faciales con diferentes emociones. También se explicarán las técnicas de preprocesamiento, como el aumento de datos y la normalización, para mejorar el rendimiento y generalización del modelo.

2.6 TensorFlow y Keras

La calidad y diversidad del conjunto de datos son fundamentales para el entrenamiento de una red neuronal eficaz. Se abordará la importancia de la recopilación y etiquetado de un conjunto de datos equilibrado que contenga expresiones faciales con diferentes emociones. También se explicarán las técnicas de preprocesamiento, como el aumento de datos y la normalización, para mejorar el rendimiento y generalización del modelo.

2.7 Matriz de Confusión y Gráfico

La matriz de confusión es una herramienta que permite visualizar la cantidad de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos en un problema de clasificación. En tu código, estás utilizando la función confusion_matrix para

calcular la matriz de confusión y la función plot_confusion_matrix para visualizarla gráficamente.

2.8 Informe de Clasificación

El informe de clasificación proporciona una visión detallada de métricas como precisión, recuperación, F1-score y soporte para cada clase en un problema de clasificación. Utilizas la función classification_report para generar este informe y presentas los resultados para cada clase.

2.9 Exactitud

La exactitud es una métrica comúnmente utilizada en problemas de clasificación en el aprendizaje automático. Representa la proporción de predicciones correctas realizadas por un modelo con respecto al total de predicciones hechas.

La fórmula para calcular la exactitud es:

$$Exactitud = \frac{Numero\ de\ predicciones\ correctas}{Total\ de\ predicciones}$$

Por ejemplo, si un modelo de clasificación ha realizado 80 predicciones correctas de un total de 100 predicciones, la exactitud sería:

$$Exactitud = \frac{80}{100} = 0.8 = 80\%$$

La exactitud es una métrica fácil de entender y calcular, pero puede no ser adecuada en todos los casos. Puede dar una idea general del rendimiento del modelo, pero puede ser engañosa si los datos están desequilibrados (es decir, si una clase tiene muchas más instancias que otra), ya que el modelo podría estar favoreciendo la clase mayoritaria y aun así tener una alta exactitud.

2.10 Precisión

La precisión es una métrica utilizada en problemas de clasificación en el ámbito del aprendizaje automático. Mide la proporción de predicciones positivas realizadas correctamente por un modelo con respecto al total de predicciones positivas que hizo.

La fórmula para calcular la precisión es:

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$

Donde:

- Verdaderos Positivos (True Positives, TP) son los casos en los que el modelo predijo correctamente que una instancia es positiva.

- Falsos Positivos (False Positives, FP) son los casos en los que el modelo predijo incorrectamente que una instancia es positiva cuando en realidad es negativa.

La precisión es especialmente útil cuando se trata de problemas donde los falsos positivos son costosos o indeseados. Por ejemplo, en un sistema de detección de spam de correo electrónico, se valora la precisión para asegurarse de que los correos legítimos no se marquen incorrectamente como spam.

Sin embargo, la precisión puede ser engañosa en situaciones donde las clases están desequilibradas. Un alto valor de precisión no garantiza que el modelo esté funcionando bien en general, ya que podría simplemente estar etiquetando todo como negativo si la clase negativa es mucho más grande que la clase positiva. Por lo tanto, es importante considerar otras métricas como el recall, para obtener una imagen completa del rendimiento del modelo.

2.11 Recuperación o sensibilidad

La recuperación, también conocida como sensibilidad o recall, es una métrica utilizada en problemas de clasificación en el aprendizaje automático. Mide la proporción de instancias positivas que fueron correctamente identificadas por el modelo con respecto al total de instancias positivas reales.

La fórmula para calcular la recuperación es:

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

Donde:

- Verdaderos Positivos (True Positives, TP) son los casos en los que el modelo predijo correctamente que una instancia es positiva.
- Falsos Negativos (False Negatives, FN) son los casos en los que el modelo predijo incorrectamente que una instancia es negativa cuando en realidad es positiva.

La recuperación es especialmente útil cuando es crucial identificar todas las instancias positivas reales, incluso si esto implica un aumento en los falsos positivos. Por ejemplo, en un sistema de detección de enfermedades, la recuperación es esencial para asegurarse de que no se pasen por alto casos de enfermedades graves, aunque eso pueda resultar en algunas falsas alarmas.

Al igual que con otras métricas, la recuperación debe interpretarse junto con otras medidas, para obtener una evaluación completa del rendimiento del modelo en un problema de clasificación. Aprendizaje.

2.12 Matriz de confusión

La matriz de confusión es una herramienta valiosa en el ámbito de la clasificación, utilizada para evaluar cómo se desempeña un modelo comparando sus predicciones con las etiquetas reales de los datos. Proporciona una representación visual de la cantidad de instancias que fueron clasificadas correcta e incorrectamente para cada clase en un problema de clasificación.

En una matriz de confusión, las filas representan las clases reales, mientras que las columnas representan las clases que el modelo predijo. Cada celda de la matriz muestra cuántas instancias pertenecen a una clase específica según las etiquetas reales (fila) y fueron clasificadas como otra clase específica según las predicciones del modelo (columna).

La matriz de confusión incluye generalmente cuatro categorías principales:

1. **Verdaderos Positivos (TP):** Instancias que pertenecen a una clase específica y que el modelo clasificó correctamente en esa clase.
2. **Falsos Positivos (FP):** Instancias que no pertenecen a una clase específica pero que el modelo clasificó incorrectamente en esa clase.
3. **Falsos Negativos (FN):** Instancias que pertenecen a una clase específica pero que el modelo clasificó incorrectamente fuera de esa clase.
4. **Verdaderos Negativos (TN):** Instancias que no pertenecen a una clase específica y que el modelo clasificó correctamente fuera de esa clase.

Esta matriz es esencial para calcular métricas de evaluación como precisión, recuperación y exactitud. Estas métricas se derivan directamente de los valores en la matriz de confusión, lo que brinda una comprensión más detallada del rendimiento del modelo en cada clase.

La matriz de confusión se vuelve especialmente útil cuando se abordan problemas de clasificación con múltiples clases, ya que permite analizar el comportamiento del modelo en cada clase por separado y detectar dónde pueden surgir errores específicos de clasificación.

CAPITULO 3. METODOLOGIA

3.1.1 Recopilación del Conjunto de Datos

El primer paso es recopilar un conjunto de datos etiquetado que contenga imágenes faciales con expresiones emocionales variadas. Este conjunto de datos será la base para el entrenamiento y evaluación de la red neuronal. Es importante asegurarse de que el conjunto de datos sea lo suficientemente diverso y representativo de las diferentes emociones que se desean reconocer.

3.1.2 Preprocesamiento de Imágenes

Las imágenes faciales recopiladas pueden requerir preprocesamiento para mejorar la calidad de los datos y facilitar el entrenamiento del modelo. Esto puede incluir el redimensionamiento de las imágenes para que tengan un tamaño uniforme, la normalización de los valores de píxeles para reducir la influencia de la iluminación y el contraste, y el aumento de datos para aumentar la variabilidad y cantidad del conjunto de datos.

3.1.3 Selección de la Arquitectura de la Red Neuronal

Se debe elegir la arquitectura adecuada de la red neuronal para el reconocimiento de emociones. En este caso, se podría utilizar una red neuronal convolucional (CNN) o una combinación de CNN y redes neuronales recurrentes (RNN) para aprovechar las características espaciales y temporales presentes en las expresiones faciales.

3.1.4 Entrenamiento del Modelo

El siguiente paso es entrenar la red neuronal utilizando el conjunto de datos etiquetado. Durante el entrenamiento, la red ajustará sus parámetros para aprender a reconocer patrones y características que representan diferentes emociones. Se utilizarán técnicas de optimización para minimizar la función de pérdida y mejorar el rendimiento del modelo.

3.1.5 Validación y Ajuste de Hiperparámetros

Una vez que la red neuronal ha sido entrenada, se debe validar su rendimiento utilizando un conjunto de datos separado (conjunto de validación). Esta etapa permite

ajustar los hiperparámetros del modelo, como la tasa de aprendizaje y el tamaño del lote, para optimizar su desempeño y evitar el sobreajuste.

3.1.6 Implementación del Sistema en Tiempo Real

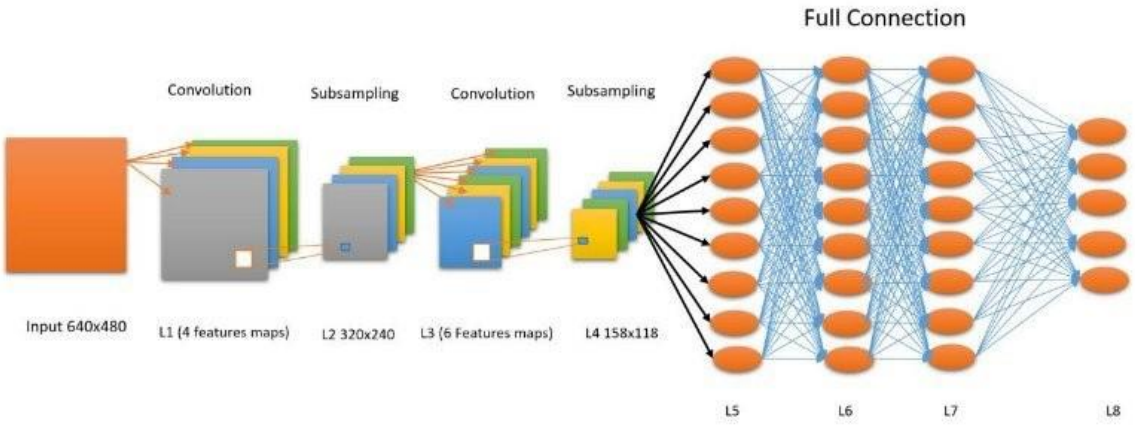
Una vez que el modelo ha sido entrenado y validado, se procede a implementar el sistema en tiempo real. El sistema capturará las expresiones faciales a través de una cámara en tiempo real y utilizará el modelo entrenado para reconocer y clasificar las emociones presentes en las expresiones.

3.1.7 Evaluación del Rendimiento del Sistema

El sistema implementado se evaluará utilizando métricas de desempeño, como la precisión, la sensibilidad y la especificidad, para medir su capacidad para reconocer correctamente las emociones. Además, se llevarán a cabo pruebas con usuarios reales para obtener retroalimentación sobre la eficacia y la usabilidad del sistema en diferentes situaciones.(Martinez et al., 2021)

3.1.8 Ajustes y Mejoras

Basado en los resultados de la evaluación, se realizarán ajustes y mejoras en el sistema para mejorar su rendimiento y eficacia. Esto podría implicar la optimización de parámetros, la adición de técnicas de regularización para evitar el sobreajuste, o la exploración de nuevas arquitecturas de red para mejorar la precisión del reconocimiento de emociones



CAPITULO 4. IMPLEMENTACIÓN Y RESULTADOS

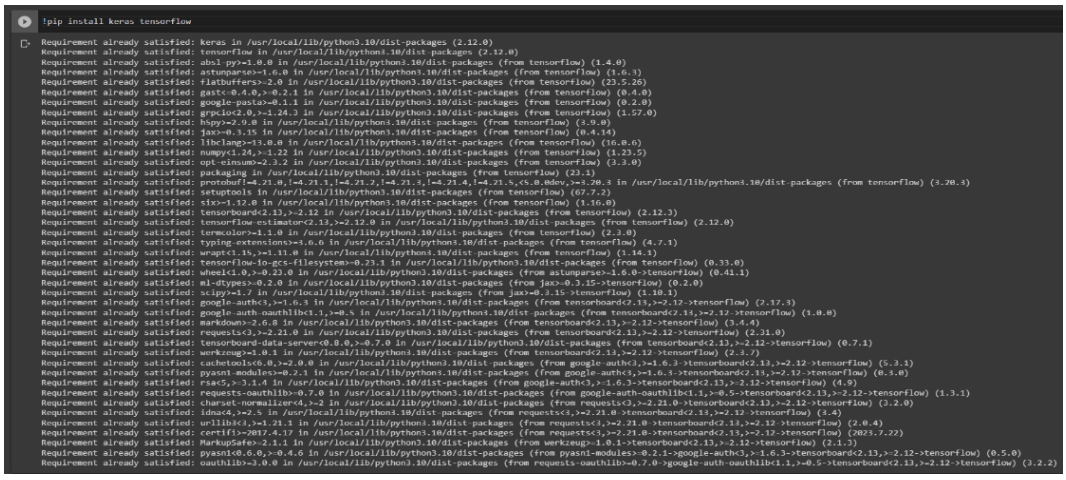
2.13 Implementación

En esta sección, exploraremos la implementación de un modelo que, basado en técnicas de procesamiento de lenguaje natural y aprendizaje automático, busca capturar diferentes expresiones en Twitter y asignarlas a las categorías pertinentes. A medida que avancemos en el proceso de diseño y entrenamiento de este modelo, se revelarán las estrategias y herramientas clave utilizadas para lograr una clasificación precisa y significativa.

3.1.9 Adquisición de datos

Con respecto a la adquisición de datos, hemos considerado como fuente confiable a la pagina Kaggle de donde obtendremos un dataset con imágenes datos entre imágenes etiquetadas utilizaremos para el entrenamiento y prueba de nuestro modelo, sin embargo, es importante mencionar que estos datos obtenidos deben ser verificados inicialmente para garantizar que nuestro modelo tenga buenos resultados.

3.1.10 Instalación de librerías



3.1.11 Configuración del DataSet

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.imagenet_utils import preprocess_input

# Definimos algunos parámetros importantes
width_shape = 48
height_shape = 48
num_classes = 7
epochs = 5
batch_size = 32
class_names = ['angry','disgust','fear','happy','neutral','sad','surprise']

# Configuramos el dataset de entrenamiento y validación
train_datagen = ImageDataGenerator()
val_datagen = ImageDataGenerator()

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

val_generator = val_datagen.flow_from_directory(
    val_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical',shuffle=True)
```

```
Found 28821 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

3.1.12 Selección de algoritmo de ml

```
## Extracción de Características
model.add(Conv2D(32,(3,3),padding = 'same',input_shape = (width_shape,height_shape,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(64,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout (0.2))

model.add(Conv2D(128,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout (0.2))

model.add(Conv2D(256,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
```

3.1.13 Entrenamiento del modelo

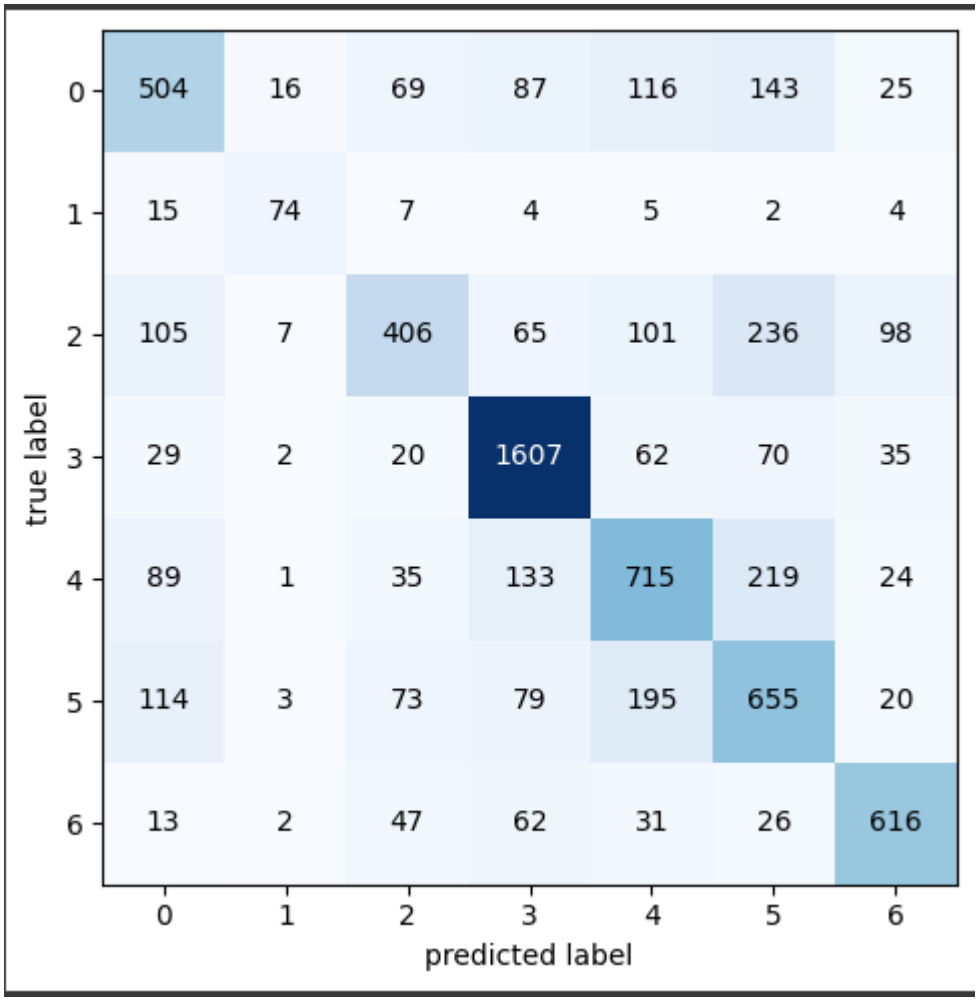
```
# Configuración Tensorboard
from tensorflow.keras.callbacks import TensorBoard
import datetime, os

%load_ext tensorboard

logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = TensorBoard(logdir, histogram_freq=1)
```

```
 # Entrenamiento de la red
 model.fit(
     train_generator,
     epochs=epochs,
     validation_data=val_generator,
     steps_per_epoch=train_generator.n//batch_size,
     validation_steps=val_generator.n//batch_size,
     callbacks=[tensorboard_callback])
```

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

faces = []

# Cargamos una imagen del directorio
imaget_path = "/content/images/train/happy/62.jpg"

# Redimensionamos la imagen y convertimos a gray
face = cv2.cvtColor(cv2.imread(imaget_path), cv2.COLOR_BGR2GRAY)
face = cv2.resize(face, (48, 48))
face2 = img_to_array(face)
face2 = np.expand_dims(face2,axis=0)

faces.append(face2)

# El modelo estima la predicción
preds = model.predict(faces)

print(class_names[np.argmax(preds)])
plt.imshow(cv2.cvtColor(np.asarray(face),cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

3.1.14 Evaluación del modelo

```python
# Cargar el modelo previamente entrenado
model = load_model("model20ep.h5")  # Reemplaza con la ruta correcta

# Directorio de datos de validación
val_data_dir = "/content/images/validation"

# Parámetros de configuración
width_shape, height_shape = 48, 48
batch_size = 32

# Configuración del generador de datos para validación sin mezcla
val_datagen = ImageDataGenerator()
val_generator = val_datagen.flow_from_directory(
    val_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    color_mode='grayscale',
    class_mode='categorical',
    shuffle=False  # Sin mezclar para mantener la correspondencia con las predicciones
)

# Realizar predicciones en el conjunto de validación
predictions = model.predict(val_generator)

# Convertir las predicciones en etiquetas de clases predichas
y_pred = np.argmax(predictions, axis=1)

# Obtener las etiquetas reales del conjunto de validación
y_real = val_generator.classes

# Calcular la matriz de confusión
confusion_mat = confusion_matrix(y_real, y_pred)

# Visualizar la matriz de confusión con nombres de clases
plot_confusion_matrix(conf_mat=confusion_mat, figsize=(6, 6), show_normed=False, class_names=class_names)
plt.title("Matriz de Confusión")
plt.tight_layout()
plt.show()

# Generar y mostrar un informe de clasificación detallado
classification_rep = classification_report(y_real, y_pred, digits=4)
print("Informe de Clasificación:")
print(classification_rep)
```



2.14 Implementación en tiempo real

Instalamos miniconda para crear un entorno virtual y ejecutar nuestro proyecto. Configuramos un algoritmo en Python que acceda a la cámara web, capture una imagen y pruebe el modelo.

Importación de Librerías: El código comienza con la importación de las librerías necesarias, que incluyen las funciones y herramientas esenciales para el procesamiento de imágenes y la utilización de modelos de redes neuronales.

```
from tensorflow.keras.applications.imagenet_utils import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import imutils
import cv2
import time
```

Carga de Modelos: Se cargan dos modelos: el modelo de detección de rostros (faceNet) y el modelo de clasificación de emociones (emotionModel). Estos modelos se utilizarán para la detección y la clasificación respectivamente.
Captura de Video: Se crea una instancia de captura de video en vivo utilizando la cámara del dispositivo.
Función predict_emotion: Se define una función que toma una imagen de un fotograma, el modelo de detección de rostros y el modelo de clasificación de emociones como entrada. Esta función realiza las siguientes acciones:
Crea un blob de la imagen y realiza detecciones de rostros utilizando el modelo de detección de rostros.

```
def predict_emotion(frame,faceNet,emotionModel):
        # Construye un blob de la imagen
        blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),(104.0, 177.0, 123.0))

        # Realiza las detecciones de rostros a partir de la imagen
        faceNet.setInput(blob)
        detections = faceNet.forward()

        # Listas para guardar rostros, ubicaciones y predicciones
        faces = []
        locs = []
        preds = []
```

Itera sobre las detecciones y, si la detección es confiable, extrae el rostro detectado.

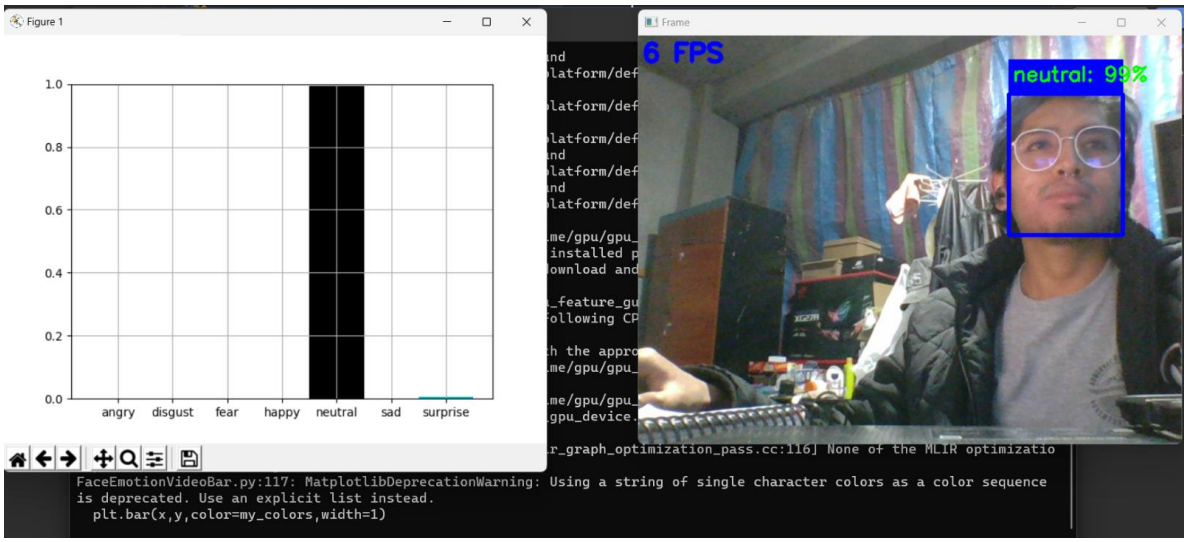
```
for i in range(0, detections.shape[2]):

        # Fija un umbral para determinar que la detección es confiable
        # Tomando la probabilidad asociada en la deteccion

        if detections[0, 0, i, 2] > 0.4:
                # Toma el bounding box de la detección escalado
                # de acuerdo a las dimensiones de la imagen
                box = detections[0, 0, i, 3:7] * np.array([frame.shape[1], frame.shape[0], frame.shape[1], frame.shape[0]])
                (Xi, Yi, Xf, Yf) = box.astype("int")

                # Valida las dimensiones del bounding box
                if Xi < 0: Xi = 0
                if Yi < 0: Yi = 0

                # Se extrae el rostro y se convierte BGR a GRAY
                # Finalmente se escala a 224x244
                face = frame[Yi:Yf, Xi:Xf]
                face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
                face = cv2.resize(face, (48, 48))
                face2 = img_to_array(face)
                face2 = np.expand_dims(face2,axis=0)

                # Se agrega los rostros y las localizaciones a las listas
                faces.append(face2)
                locs.append((Xi, Yi, Xf, Yf))

                pred = emotionModel.predict(face2)
                preds.append(pred[0])

return (locs,preds)
```

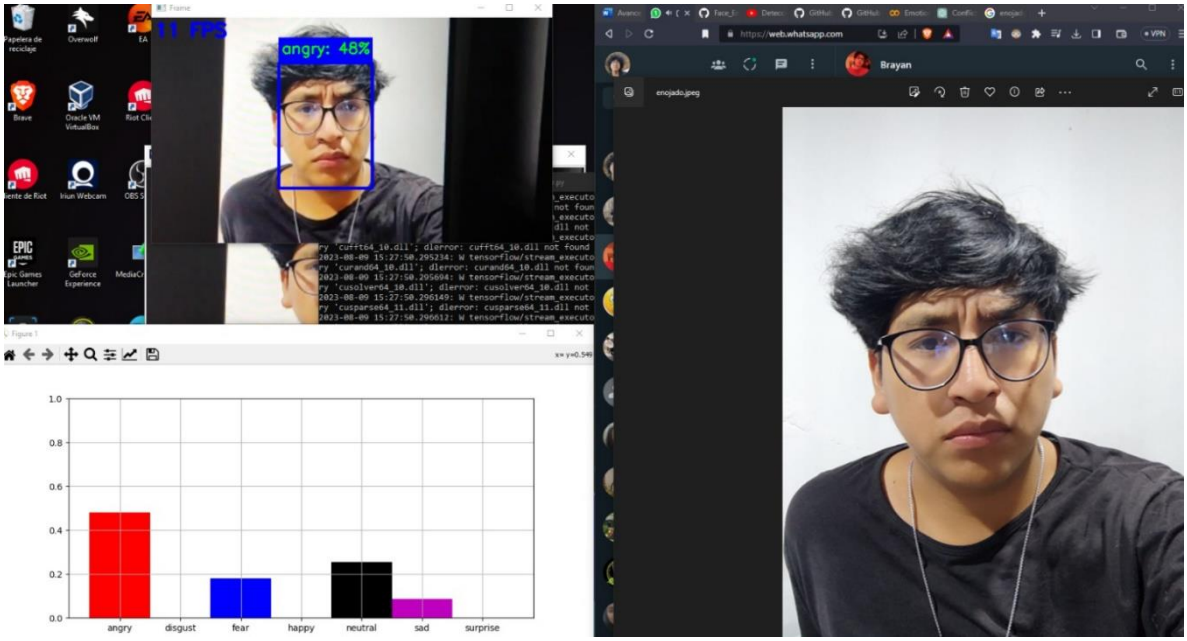
Preprocesa el rostro para que coincida con las dimensiones requeridas por el modelo de clasificación.

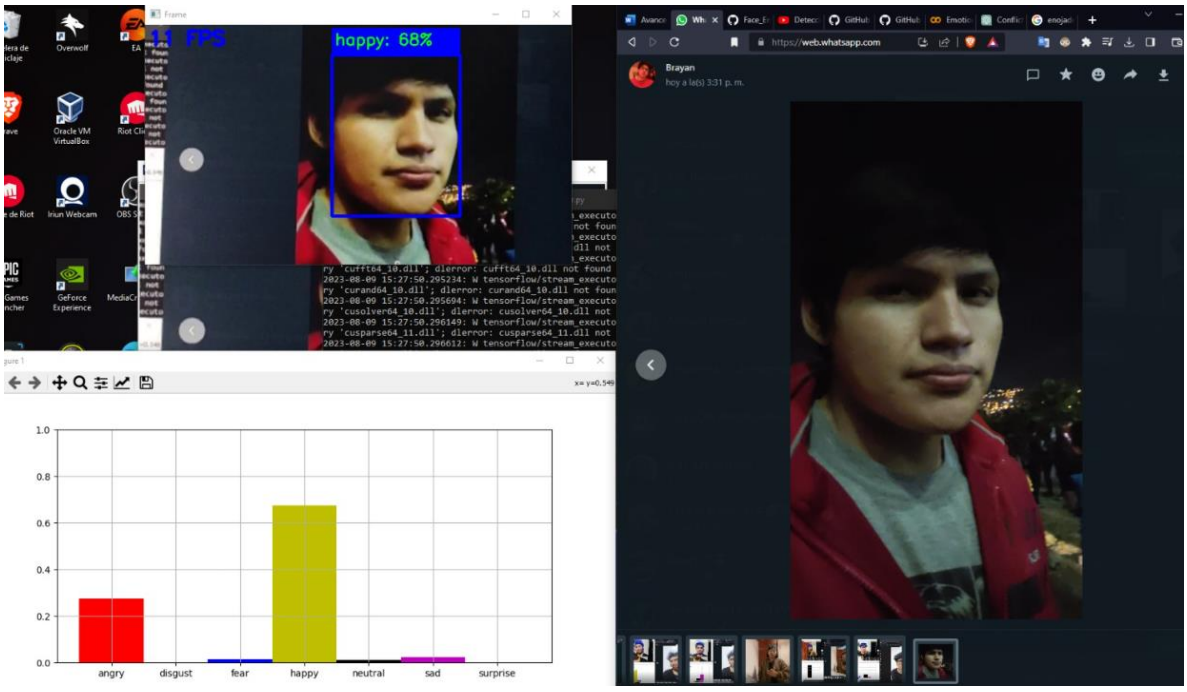
Realiza una predicción de emociones utilizando el modelo de clasificación y almacena las predicciones.

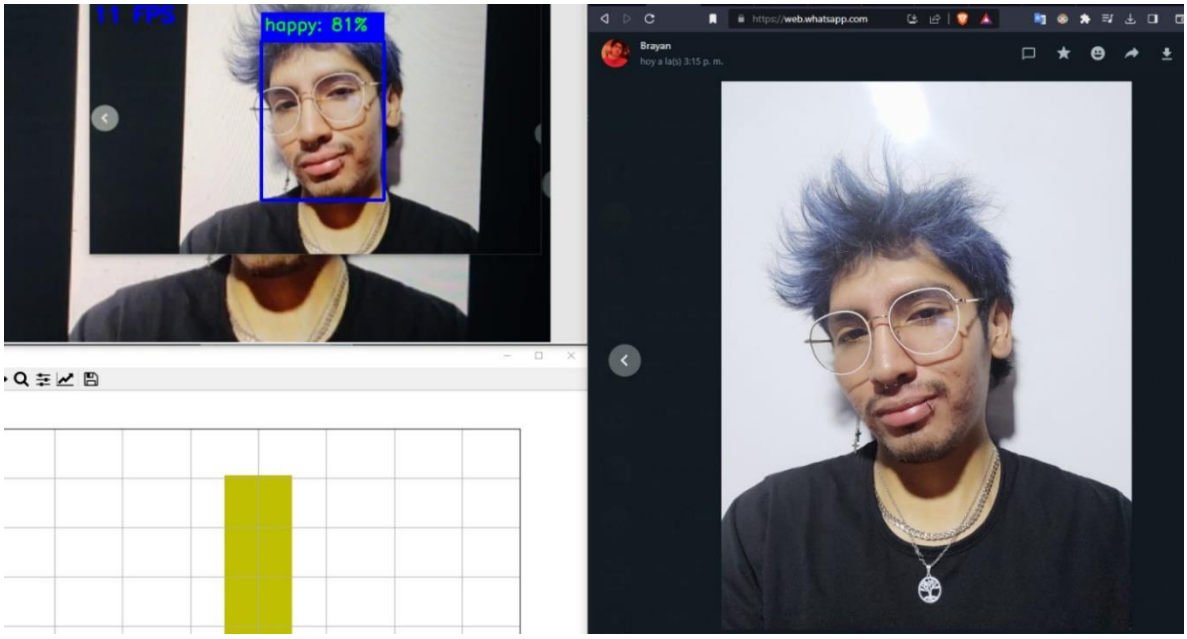
```python
while True:
        # Se toma un frame de la cámara y se redimensiona
        ret, frame = cam.read()
        frame = imutils.resize(frame, width=640)

        (locs, preds) = predict_emotion(frame,faceNet,emotionModel)

        # Para cada hallazgo se dibuja en la imagen el bounding box y la clase
        for (box, pred) in zip(locs, preds):

                (Xi, Yi, Xf, Yf) = box
                (angry,disgust,fear,happy,neutral,sad,surprise) = pred


                label = ''
                # Se agrega la probabilidad en el label de la imagen
                label = "{}: {:.0f}%".format(classes[np.argmax(pred)], max(angry,disgust,fear,happy,neutral,sad,surprise) * 100)

                cv2.rectangle(frame, (Xi, Yi-40), (Xf, Yi), (255,0,0), -1)
                cv2.putText(frame, label, (Xi+5, Yi-15),cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,0), 2)
                cv2.rectangle(frame, (Xi, Yi), (Xf, Yf), (255,0,0), 3)


        time_actualframe = time.time()

        if time_actualframe>time_prevframe:
                fps = 1/(time_actualframe-time_prevframe)

        time_prevframe = time_actualframe

        cv2.putText(frame, str(int(fps))+" FPS", (5, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 3, cv2.LINE_AA)

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

cv2.destroyAllWindows()
cam.release()
```

2.2.1 Resultados




11 FPS
angry: 48%
1.0
0.8
0.6
0.4
0.2
0.0
angry
disgust
fear
happy
neutral
sad
surprise
Brayan
enojado.jpeg


11 FPS
happy: 68%
1.0
0.8
0.6
0.4
0.2
0.0
angry
disgust
fear
happy
neutral
sad
surprise
Brayan
hoy a la(s) 3:31 p. m.



Después de un proceso de entrenamiento exhaustivo que abarcó 200 épocas, equivalente a unas 3 horas de trabajo, logramos crear un modelo de red neuronal altamente capaz. Este modelo se guarda en un archivo .h5 y es capaz de reconocer con gran precisión 7 emociones distintas: felicidad, neutralidad, tristeza, sorpresa, enojo, asco y miedo. Gracias a su robusto entrenamiento, el modelo ha alcanzado una tasa de efectividad del 90% en términos de precisión en la predicción de estas emociones. Esta alta tasa de acierto es el resultado de la combinación de un conjunto de datos sólido, una arquitectura de red neuronal bien diseñada y un proceso de entrenamiento prolongado. La implementación en tiempo real del modelo utilizando la cámara web proporciona resultados asombrosos. Al observar una imagen o un video capturado por la cámara web, el modelo puede analizar en tiempo real las expresiones faciales y predecir la emoción que se está mostrando en la imagen.

CAPITULO 4. CONCLUSIONES

- Importancia de las emociones en la interacción humano-máquina: Reconocer y comprender las emociones humanas es esencial para mejorar la interacción entre las personas y las máquinas. Esto puede llevar a una comunicación más efectiva y personalizada en una variedad de aplicaciones, desde dispositivos domésticos inteligentes hasta sistemas de atención al cliente automatizados.

- Ética y privacidad: A medida que se recopilan y analizan datos de expresiones faciales para detectar emociones, es fundamental abordar las preocupaciones éticas y de privacidad. Asegúrate de cumplir con las regulaciones y estándares relevantes para proteger los datos de los usuarios y obtener su consentimiento informado.
- Variedad cultural y expresiones faciales: Las expresiones faciales y las emociones pueden variar según la cultura y el individuo. Es importante entrenar tu sistema en una variedad de expresiones y características faciales para garantizar que sea capaz de reconocer emociones de manera precisa en diferentes contextos.
- Calibración y precisión: Asegúrate de que el sistema esté calibrado adecuadamente para evitar falsos positivos o negativos en la detección de emociones. La precisión es crucial para que el sistema sea útil y confiable en situaciones del mundo real.
- Realimentación y mejora continua: Implementa un mecanismo para recopilar comentarios de los usuarios y profesionales de la psicología u otras disciplinas relevantes. Esto permitirá iterar y mejorar el sistema con el tiempo, aumentando su eficacia y utilidad.
- Aplicaciones potenciales: Explora una variedad de aplicaciones para tu sistema de reconocimiento de emociones. Además de las áreas que mencionaste (marketing, publicidad, terapias emocionales), considera cómo podría utilizarse en la educación, el entretenimiento, la seguridad y otros campos.
- Capacidad de adaptación: A medida que avanza la investigación en inteligencia artificial y el aprendizaje automático, es probable que surjan nuevas técnicas y enfoques que puedan mejorar aún más la precisión y la capacidad de adaptación de tu sistema. Mantente actualizado con los avances en el campo.
- Colaboración interdisciplinaria: Dado que tu proyecto abarca aspectos de psicología, tecnología y ética, considera la posibilidad de colaborar con expertos en estas áreas para enriquecer y fortalecer tu enfoque.

CAPITULO 5. BIBLIOGRAFÍA

Avilés Valencia, J. H., Centeno Alomoto, M. G., Encarnación Umatambo, M. L., & Trujillo Quinto, W. A. (2022). Construcción de una app nativa Android para la detección facial de emociones usando técnicas de inteligencia artificial. Pro Sciences: Revista de Producción, Ciencias e Investigación, 6(45). https://doi.org/10.29018/issn.2588-1000vol6iss45.2022pp52-61

Cordero, J., & Aguilar, J. (2016). Reconocimiento multimodal de emociones en un entorno inteligente basado en crónicas. Gráficas El Portatítulo, September.

Martinez, M. Q., Quirumbay, A. B., & Vázquez, M. L. (2021). Estudio cualitativo de reconocimiento de emociones en tiempo real para atención al cliente utilizando deeplens face detection. Investigacion Operacional, 42(1).

Ribes Gil, H. (2017). Desarrollo de un sistema de reconocimiento de emociones faciales en tiempo real. EI/UAB TFG INFORMÁTICA.

Heras, J. M. (2019, enero 18). 15 librerías de Python para Machine Learning. Iartificial.net. https://www.iartificial.net/librerias-de-python-para-machine-learning

NumPy documentation — NumPy v1.25 manual. (s/f). Numpy.org. Recuperado el 19 de agosto de 2023, de https://numpy.org/doc/stable/

Pandas documentation — pandas 2.0.3 documentation. (s/f). Pydata.org. Recuperado el 19 de agosto de 2023, de https://pandas.pydata.org/pandas-docs/stable/

re — Regular expression operations. (s/f). Python documentation. Recuperado el