Instituto Tecnológico y de Estudios Superiores de Monterrey

Implementación de un modelo de Deep Learning.

Integrantes:

Carlos David Lozano Sanguino A01275316

Monterrey, Nuevo León, México 4 de Noviembre del 2023

Inteligencia artificial avanzada para la ciencia de datos II (Gpo 501)

Introducción

A lo largo de este periodo, hemos dedicado un esfuerzo continuo en la mejora de nuestros modelos previamente entrenados con el objetivo de crear una aplicación de asistencia y registro de participación basada en el reconocimiento facial. Esta iniciativa ha sido desarrollada en colaboración con nuestro socio formador y se ha enfocado en ofrecer un modelo integral que simplifique las tareas diarias de profesores y la gestión del registro de asistencia en entornos educativos, como universidades y escuelas.

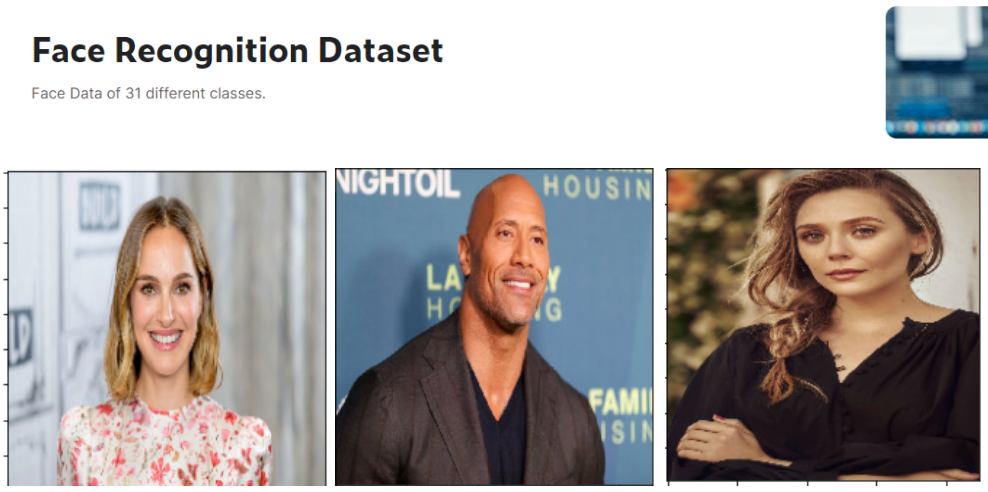
Como parte de nuestro compromiso adicional, también hemos implementado un modelo de reconocimiento facial similar al que estamos desarrollando, que sirve como evidencia de nuestro aprendizaje y conocimiento adquirido a lo largo de esta concentración. Nuestro objetivo es demostrar la aplicabilidad práctica de las habilidades y técnicas que hemos adquirido en este período de formación.

Selección del Modelo

El problema de reconocimiento facial en asistencia en aulas involucra la identificación precisa de individuos en imágenes o secuencias de vídeo, a menudo en tiempo real. Las CNN se han destacado en tareas de visión por computadora, incluido el reconocimiento facial, debido a su capacidad inherente para extraer características relevantes de las imágenes. Es por esto que elegí implementar para este entregable una Red Neuronal Convolucional en base a los ejemplos que vimos en clase los cuales estaban para los datos MNIST pero aplicado en este caso para el reconocimiento facial.

Selección del Dataset

Para este modelo en específico usamos un dataset de acceso público y libre que contenía las caras de personas famosos como henry cavill, robert downey jr, etc con el fin de entrenar el modelo para el reconocimiento facial de estas personas y asi usarlo posteriormente con nuevas caras de estudiante o cualquier persona. Este Dataset puede encontrarse en kaggle como Face Recognition Dataset con una cantidad total de 2562 imágenes de rostros el cual concluimos que es una cantidad suficiente para el entrenamiento de los datos.



Creación del Modelo

Framework utilizado:
El framework utilizado para el modelo sería tanto tensor flow y keras para el preprocesamiento de datos el cual involucra la división del dataset y la creación de la red convolucional 2D con la funcion de activacion relu entre otras cosas ademas de multiples librerias adicionales necesarias para la implementación del modelo como ImageDataGenerator que es parte de tensorflow que sera necesario

para la division del dataset y preprocesamiento, Conv2D,MaxPooling2D,Dense,Flatten,Dropout que viene de Keras para la creación del modelo y matplotlib para la visualización de los resultados entre otras cosas.

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

Aproximación inicial:

Para la aproximación inicial o el primer intento sin ajustar los parámetros se descargo el dataset de famosos y se usó el ImageDataGenerator para reescalar los píxeles entre 0 y 1. Posteriormente se dividió el directorio de los datos en entrenamiento y prueba dejando el 80% de los datos para entrenamiento mientras que 20% para la validacion/prueba como se a continuación en el resultado de la celda:

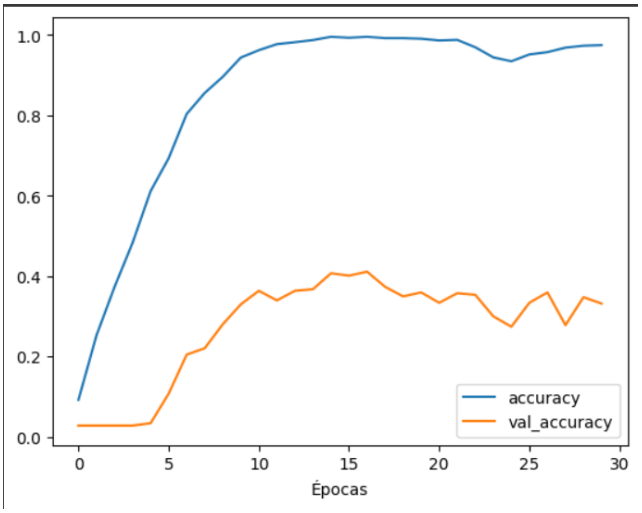
```
Found 2068 images belonging to 31 classes.
Found 504 images belonging to 31 classes.
```

Tras esto se creó el modelo aplicando la librería de Keras que trae el modelo Conv2D con la función de activación Relu que vimos en clase por su capacidad para introducir no linealidad en el modelo sin afectar en gran medida el proceso de entrenamiento. Con la creación del modelo junto a sus parámetros se entreno el modelo con la función fit con un total de 30 epochs.

```
# Entrenar el modelo
history = model.fit(train_ds, epochs=30, validation_data=val_ds, verbose=1)
```

```
Epoch 2/30
65/65 [==============================] - 54s 834ms/step - loss: 2.6711 - accuracy: 0.2524 - val_loss: 5.3472 - val_accuracy: 0.0278
Epoch 3/30
65/65 [==============================] - 49s 761ms/step - loss: 2.1322 - accuracy: 0.3728 - val_loss: 5.6695 - val_accuracy: 0.0278
Epoch 4/30
65/65 [==============================] - 43s 659ms/step - loss: 1.7422 - accuracy: 0.4826 - val_loss: 6.2949 - val_accuracy: 0.0278
Epoch 5/30
65/65 [==============================] - 44s 686ms/step - loss: 1.3489 - accuracy: 0.6112 - val_loss: 5.5463 - val_accuracy: 0.0337
Epoch 6/30
```

Tras esto se obtuvo una accuracy de validación bajo y graficamos los resultados en la siguiente gráfica.

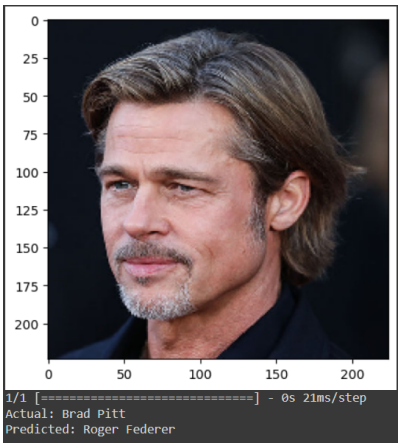


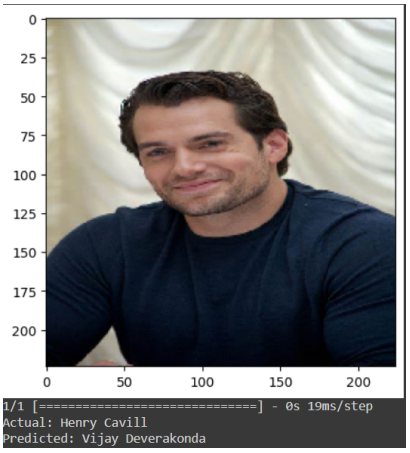
Se obtuvo un accuracy alto sin embargo solo en los datos de entrenamiento ya que esta debajo del 0.5 los de validación lo cual no es lo óptimo ya que se adaptó demasiado a los datos lo cual no podrá generar al 100% bien la predicción de datos o reconocimiento de las caras de famosos.

Para confirmar lo anterior se imprimió en algunas predicciones en la consola para ver si reconoce las caras de los actores o no los hace confundiendo al actor. Con la siguiente función se puede obtener la imagen y datos de la cara de la persona donde el programa podrá decir si es tal persona y su nombre:

```
def predict_image(image_path):
    img = image.load_img(image_path, target_size=(224,224,3))
    plt.imshow(img)
    plt.show()
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    pred = model.predict(images, batch_size=32)
    print("Actual: "+(image_path.split("/")[-1]).split("_")[0])
    print("Predicted: "+classes[np.argmax(pred)])
```

Predicciones





Podemos observar que la anotación anterior fue correcta pues traía el nombre actual de la persona y su imagen pero la predicción es errónea en algunas caras por lo que es necesario realizar ajustes en los parámetros para un mejor desempeño

Segunda Aproximación:

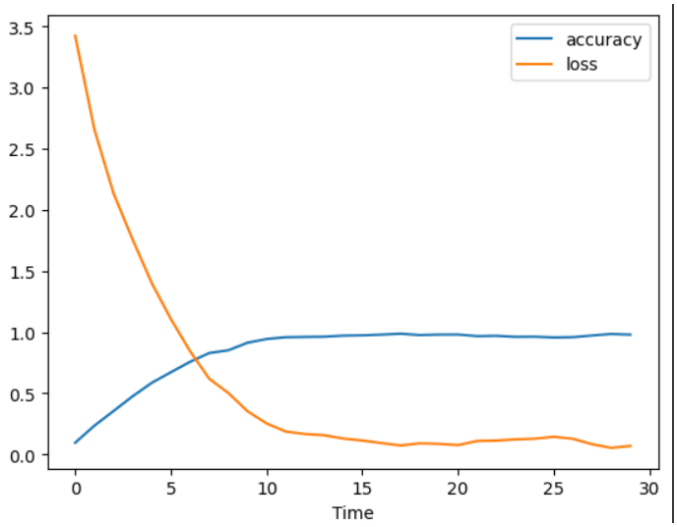
Tras haber visto el desempeño anterior decidí implementar unos cambios para mejorar el rendimiento en base a lo que hemos aprendido primero se utilizó la biblioteca ImageDataGenerator de Keras para crear un generador de datos a partir de un directorio que contiene imágenes. Estas imágenes se redimensionan a un tamaño de 224x224 píxeles y se agruparon en lotes de 32, este proceso permitió mejorar la división de datos de manera automática en vez de hacer por pasos y una manera más manual o por código la división y escalamiento de datos. Además, se obtuvieron las clases a partir de los nombres de las subcarpetas en el directorio. Este proceso permite cargar y preparar datos de imágenes de manera eficiente para su posterior uso en un modelo de aprendizaje profundo.

```
train_dir="/content/drive/MyDrive/face-recognition-dataset/Original Images/Original Images"
generator = ImageDataGenerator()
train_ds = generator.flow_from_directory(train_dir,target_size=(224, 224),batch_size=32)
classes = list(train_ds.class_indices.keys())
```

Posteriormente la creación del modelo se mantuvo igual con la misma funcion de activacion y al momento de entrenar el modelo se mantuvo la cantidad de epochs en 30 pero con un batch size 32 lo que controla la cantidad de ejemplos de datos que se utilizan en cada paso de entrenamiento utilizando 32 ejemplos de datos a la vez para calcular una actualización de los pesos del modelo. En este caso solo se utiliza la variable de entrenamiento ya que la función ImageGenerator ya cumplio su funcionalidad para el entrenamiento y validación que se verá en las pruebas

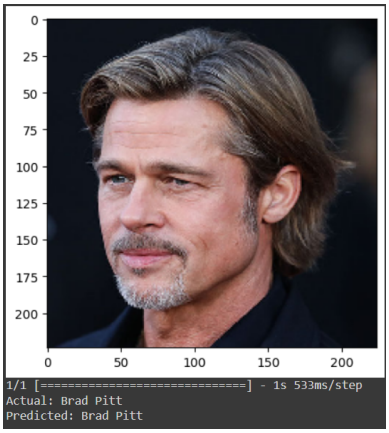
```
history = model.fit(train_ds,epochs= 30, batch_size=32)
```

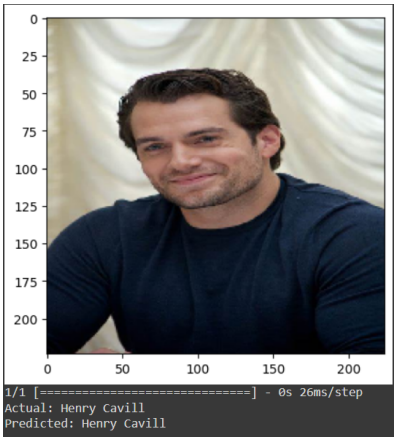
```
Epoch 2/30
81/81 [==============================] - 42s 524ms/step - loss: 2.6588 - accuracy: 0.2348
Epoch 3/30
81/81 [==============================] - 43s 536ms/step - loss: 2.1357 - accuracy: 0.3546
Epoch 4/30
81/81 [==============================] - 44s 535ms/step - loss: 1.7570 - accuracy: 0.4755
Epoch 5/30
81/81 [==============================] - 45s 556ms/step - loss: 1.4002 - accuracy: 0.5855
Epoch 6/30
81/81 [==============================] - 44s 542ms/step - loss: 1.1066 - accuracy: 0.6722
Epoch 7/30
81/81 [==============================] - 44s 539ms/step - loss: 0.8441 - accuracy: 0.7570
Epoch 8/30
81/81 [==============================] - 44s 539ms/step - loss: 0.6192 - accuracy: 0.8289
Epoch 9/30
81/81 [==============================] - 43s 533ms/step - loss: 0.5023 - accuracy: 0.8515
Epoch 10/30
81/81 [==============================] - 43s 538ms/step - loss: 0.3532 - accuracy: 0.9133
Epoch 11/30
81/81 [==============================] - 44s 544ms/step - loss: 0.2535 - accuracy: 0.9436
Epoch 12/30
81/81 [==============================] - 43s 531ms/step - loss: 0.1872 - accuracy: 0.9584
```

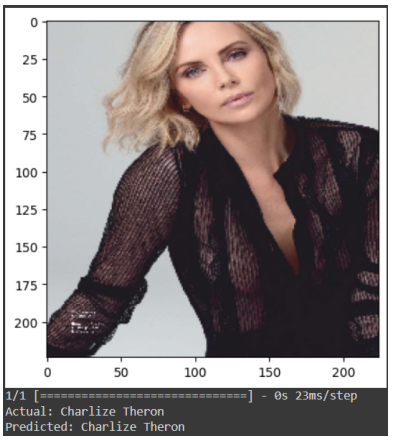


Tras esto se obtuvo un accuracy del 97% que incrementa con cada epoch mientras que la pérdida baja exponencialmente indicando un buen aprendizaje de los datos que será importante para el reconocimiento de los famosos.

Predicciones







A partir de los resultados obtenidos en las pruebas anteriores, queda claro que el modelo ha demostrado una notable capacidad para identificar a los famosos tanto en las imágenes de entrenamiento como en las imágenes de prueba, proporcionando con precisión los nombres correspondientes. Este desempeño óptimo en el reconocimiento facial en un conjunto de datos de figuras públicas se logró mediante ajustes estratégicos en los parámetros de procesamiento y entrenamiento del modelo. Estos cambios han contribuido significativamente a la mejora del rendimiento del modelo y, en última instancia, a la precisión en la identificación de rostros.

Referencias:

-Patel, V. (2020). *Face Recognition Dataset* [Data

set].https://www.kaggle.com/datasets/vasukipatel/face-recognition-dataset/data

Link del Codigo del Modelo CNN:

https://colab.research.google.com/drive/1Kb5HZNbdAmFApZpGtQI7ctAHz_XCWYFm?usp

=sharing