INFORME DEL PROYECTO FINAL DE INF-354 INTELIGENCIA ARTIFICIAL PARA EL RECONOCIMIENTO DE PERSONAS CON O SIN BARBIJO

INTRODUCCIÓN

El uso de barbijo es una medida de seguridad importante para prevenir la propagación de enfermedades, como en el caso de la pandemia de COVID-19. La implementación de un sistema automático que pueda detectar si una persona está utilizando barbijo en tiempo real puede ser de gran utilidad en entornos como hospitales, aeropuertos, estaciones de transporte público y otros lugares concurridos.

En este proyecto, utilizaremos técnicas de aprendizaje automático para entrenar un modelo que pueda analizar imágenes y realizar la clasificación de "con barbijo" y "sin barbijo". Para lograr esto, utilizaremos un conjunto de datos que incluya imágenes de personas con y sin barbijo. Utilizaremos este conjunto de datos para entrenar y evaluar nuestro modelo, ajustando los parámetros y arquitectura del modelo para obtener los mejores resultados posibles.

Una vez que el modelo esté entrenado y evaluado, podremos utilizarlo para realizar inferencias en tiempo real en imágenes o videos de personas para determinar si están usando barbijo o no. Esto podría tener aplicaciones en sistemas de monitoreo de seguridad, control de acceso en lugares públicos o privados, y otras situaciones en las que se requiera la detección automática del uso de barbijo.

1. Objetivo

El objetivo de este proyecto es desarrollar un modelo de machine learning capaz de reconocer si una persona está usando barbijo o no en imágenes o videos. El modelo se entrenará utilizando técnicas de aprendizaje automático supervisado para que pueda clasificar con precisión las imágenes o videos en dos categorías: "con barbijo" y "sin barbijo".

2. Descripción detallada de los campos del dataset

Como ya se mencionó anteriormente estamos haciendo uso de un dataset publicado en kaggle llamado “BaseDeDatosMIE820” que contiene 5000 imágenes, 2500 de personas con barbijo y 2500 sin barbijo

Entonces tenemos 2 clases para nuestros datos: Conbarbijo y SinBarbijo .

Nuestro modelo deberá identificar cada imagen según cada clase.

3. Proceso básico de análisis de datos:

3.1. Preprocesamiento de datos

El primer ajuste a los datos es la estandarización de tamaño de las imágenes a un solo tamaño de 100 x 100, esto permitirá tener un mejor control sobre las imágenes para que no varien en la entrada de los datos.

Otro ajuste de preprocesamiento que no se utiliza para el modelo, es el cambio de colores a la escala de grises de las imágenes.

3.2. Selección del clasificador

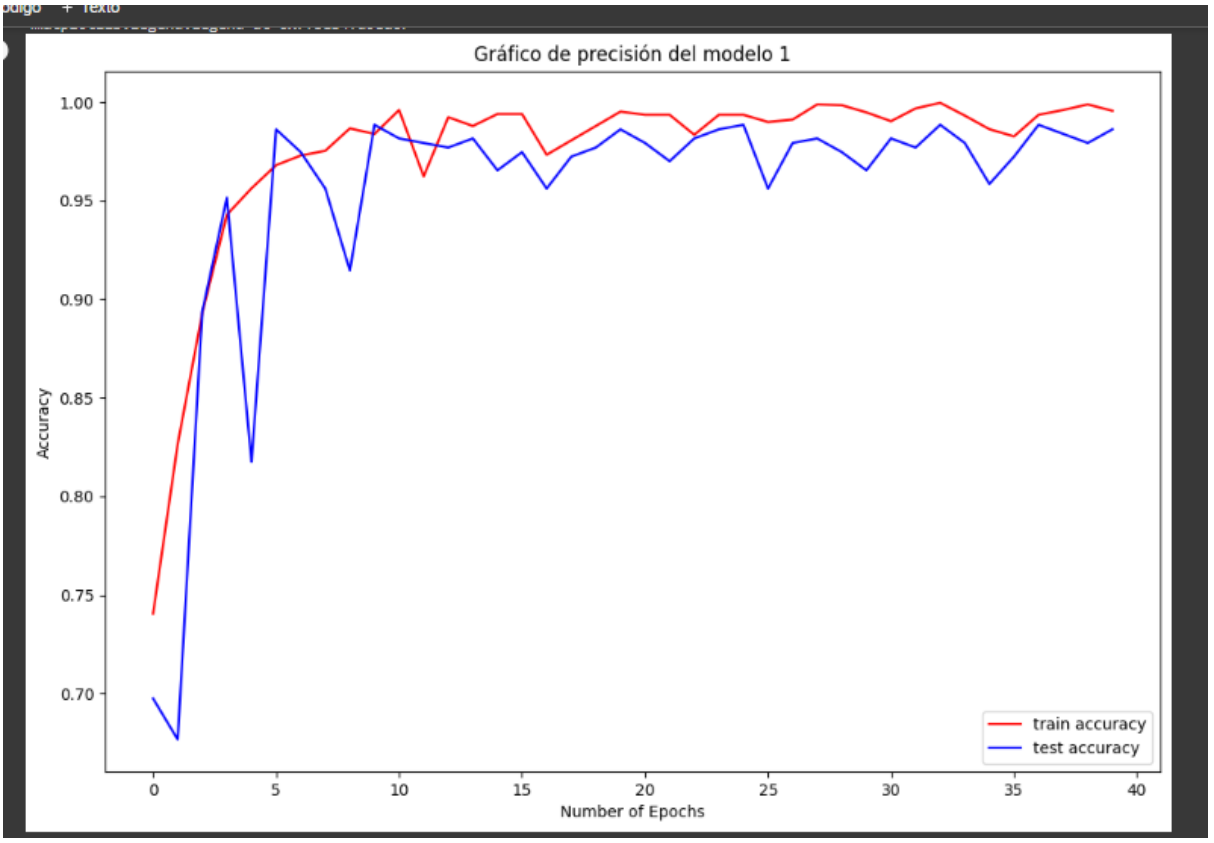
El clasificador que se ha seleccionado no tiene mucha ciencia, ya que solo necesitamos que el modelo reconozca si una persona está usando barbijo o no.

3.3. Primera ejecución del modelo:

La primera ejecución del modelo el cual es

```
model1 = Sequential([
  layers.Rescaling(1./255, input_shape=(100, 100, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Este modelo tiene 10 capas, la primera capa es de normalización, la segunda es una capa de convolución con 16 filtros, la tercera es una capa de max pooling, la cuarta es una capa de convolución con 32 filtros, la quinta es una capa de max pooling, la sexta es una capa de convolución con 64 filtros, la séptima es una capa de max pooling, la octava es una capa de aplanamiento, la novena es una capa densa con 128 neuronas y la última es una capa densa con 2 neuronas, ya que tenemos 2 clases.Despues de la compilación del modelo procede el entrenamiento al cual le damos 40 épocas y graficamos los resultados:



3.4. Splits: al menos 100 asignaciones, la mediana de la confiabilidad Académico 80(train)/20(test)

Dado nuestro conjunto de imágenes lo dividimos en conjunto de entrenamiento y validación, de acuerdo a la cantidad de imágenes que tenemos: pasamos 4000 para el entrenamiento y 1000 para la validación:

```
[ ]  train_ds = tf.keras.utils.image_dataset_from_directory(
         '/content/Rostros',
         validation_split=0.2,
         subset="training",
         seed=123,
         image_size=(100,100))
```

```
Found 5000 files belonging to 2 classes.
Using 4000 files for training.
```

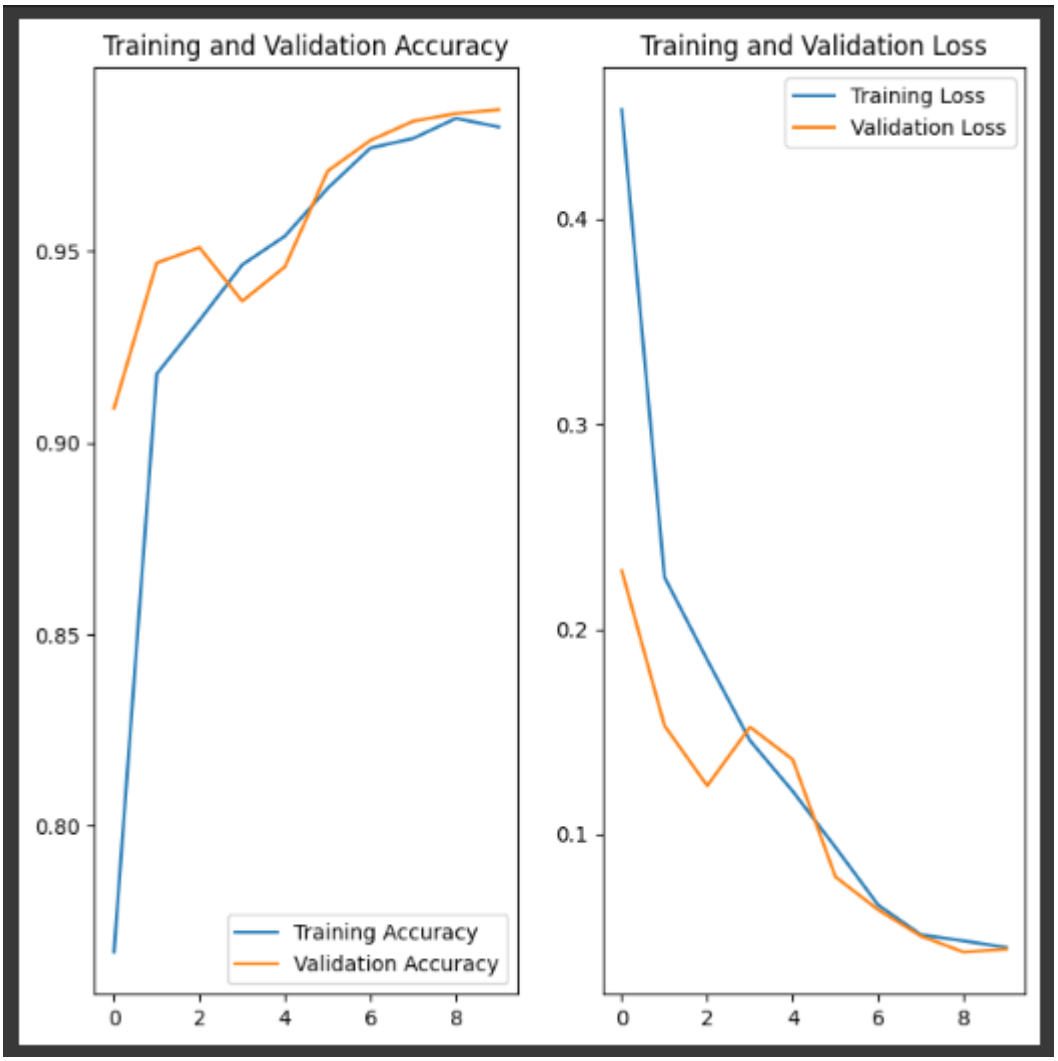
```
[ ]  val_ds = tf.keras.utils.image_dataset_from_directory(
       '/content/Rostros',
       validation_split=0.2,
       subset="validation",
       seed=200,
       image_size=(100,100))
```

```
Found 5000 files belonging to 2 classes.
Using 1000 files for validation.
```

SEGUNDO MODELO

```
#Una solución para el sobreajuste es el dropout, que consiste en eliminar
#aleatoriamente neuronas de la red, para que no se sobreajusten.
#el modelo 2 sera:

model = tf.keras.models.Sequential([
  layers.Rescaling(1./255, input_shape=(100, 100, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dropout(0.2),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Gráfica de precisión y pérdida:



Evaluación del modelo modificado.-

```
model.evaluate(val_ds,return_dict=True)
```

```
32/32 [==============================] - 1s 22ms/step - loss: 0.0277 - accuracy: 0.9890
{'loss': 0.02773580327630043, 'accuracy': 0.9890000224113464}
```

Tiene un 2 % de pérdida y 98 % de precisión

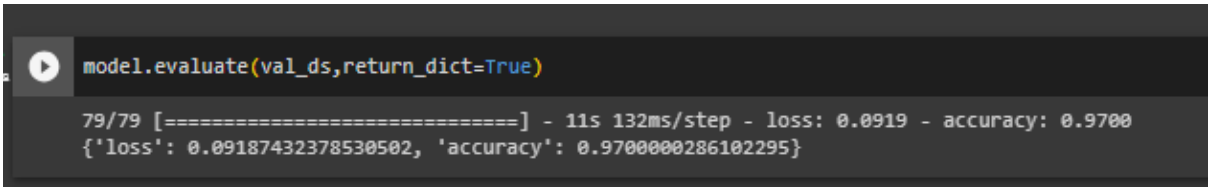
3.5. Splits: al menos 100 asignaciones INVESTIGACIÓN 50(train)/50(test)

Para ello usamos el mismo modelo, pero ahora le pasamos 50% de datos para el entrenamiento y 50% para la validación.

```
[11] train_ds = tf.keras.utils.image_dataset_from_directory(
        '/content/Rostros',
        validation_split=0.5,
        subset="training",
        seed=123,
        image_size=(100,100))

Found 5000 files belonging to 2 classes.
Using 2500 files for training.

[12] val_ds = tf.keras.utils.image_dataset_from_directory(
       '/content/Rostros',
       validation_split=0.5,
       subset="validation",
       seed=200,
       image_size=(100,100))

Found 5000 files belonging to 2 classes.
Using 2500 files for validation.
```

Luego compilamos el modelo y después realizamos el entrenamiento para 10 épocas obteniendo las siguientes gráficas de precisión y pérdida:

Podemos analizar que la curva de entrenamiento va subiendo en la precisión y bajando el porcentaje de pérdida, pero la curva de validación se aleja de la de entrenamiento , por lo que se deduce que ha bajado la precisión en comparación con la 1era ejecucion de codigo, teniendo 80% de datos para el entrenamiento y 20% para la validación.
Para evaluar qué tan preciso es el modelo en comparación del 1ero lo evaluamos de la siguiente forma:



Según la evaluación tiene un 9% de pérdida y un 97% de precisión, lo cual nos dice que ha bajado en comparación con la 1ra ejecución que tenía un 2% de pérdida y un 98% de precisión.