# PROYECTO DE INTELIGENCIA ARTIFICIAL USANDO CLASIFICACION DE IMAGENES Y REDES NEURONALES CONVOLUCIONALES PARA DETECTAR SI UNA PERSONA USA BARBIJO O NO

**RESUMEN**
Se presenta un modelo desarrollado en python colab para el reconocimiento de personas usando barbijo, esto se logra con las redes neuronales convolucioneles y la libreria de tensorflow.Se utilizo un dataset de 5000 imágenes con un 80% para el entrenamiento y 20% para la validación,además se evaluó el modelo con una matriz de confusión,llegando a obtener un 90% de exactitud.

**Palabras clave:** redes neuronales convolucionales,tensorflow

## 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 redes neuronales convolucionales 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.

### Conceptos básicos.

**Redes neuronales convolucionales**.-Una red neuronal convolucional (CNN o ConvNet) es una arquitectura de red para Deep Learning que aprende directamente a partir de datos.
Son particularmente útiles para identificar patrones en imágenes con el fin de reconocer objetos, clases y categorías. Además, pueden ser muy eficaces para clasificar datos de audio, señales y series temporales.

**Matriz de confucion.-** La matriz de confusión es un concepto relevante en el mundo de la inteligencia artificial y en el del aprendizaje automático. En esencia, se trata de una herramienta que permite analizar los resultados de cómo trabaja un algoritmo de aprendizaje supervisado. Esta matriz se presenta siempre en forma de tabla, de manera que en cada columna aparece el número de predicciones de cada clase, mientras que cada fila muestra el número real de instancias de cada clase.

**TensorFlow.-** Se trata de una librería de código libre para Machine Learning (ML). Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. TensorFlow te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos.

## DESARROLLO

### Carga del dataset

Para este proyecto usaremos un dataset publicado en la plataforma de kaggle que contiene que contiene 5000 imágenes, 2500 de personas con barbijo y 2500 sin barbijo. Entonces

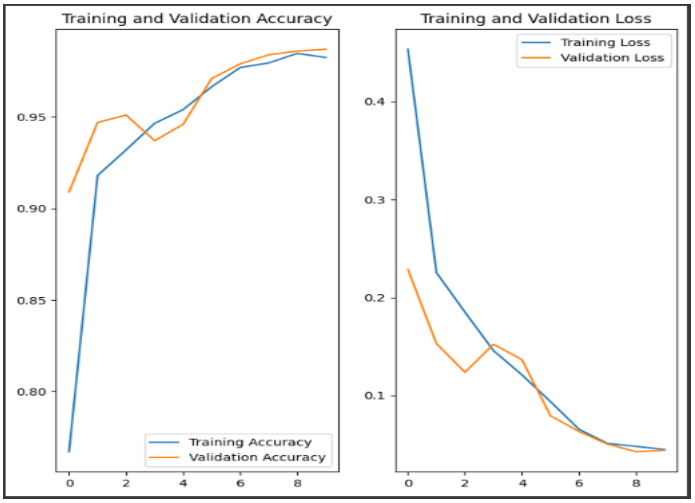
tenemos 2 clases para nuestros datos: Conbarbijo y SinBarbijo. Nuestro modelo deberá identificar cada imagen según cada clase.Además dividiremos el conjunto de datos en 80% para el entrenamiento y 20% para la validación, esto lo haremos usando la librería de tensorflow:

```
[ ]  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.

[ ]  class_names = train_ds.class_names
     print(class_names)

     ['ConMascarilla', 'SinMascarilla']
```

**Creación del modelo.-** A continuación se muestra el modelo de red convolucional

```
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)
])
```

Este modelo tiene 11 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 dropout, que consiste en eliminar aleatoriamente neuronas de la red, para que no se sobreajuste, la novena es una capa de aplanamiento, la décima es una capa densa con 128 neuronas y la última es una capa densa con 2 neuronas, ya que tenemos 2 clases.Después de la compilación del modelo procede el entrenamiento al cual le damos 10 épocas y graficamos los resultados:



Podemos analizar que la precisión de la curva de entrenamiento aumenta en cada época,así como la pérdida de información, y también se observa que la curva de entrenamiento y de validación empiezan alejados pero con cada época qué pasa están más cerca lo que es bueno porque indica que es un buen modelo.

**Evaluación del modelo.-**

Para esta parte del proyecto, necesitamos 10 imágenes al azar que no pertenezcan al conjunto de datos con el que trabajamos, para evaluar su precisión a través de una matriz de confusión que nos dirá que tan precisa es y su confiabilidad.

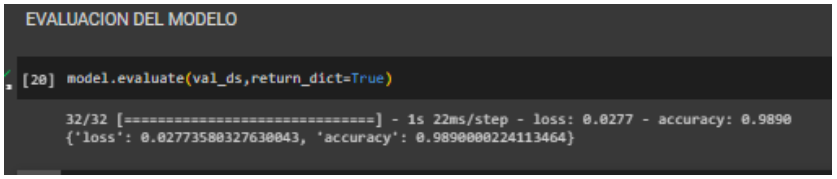

Probamos el modelo para 10 datos y este fue el resultado:

| | | Prediccion | | |
|---|---|---|---|---|
| | | Con barbijo | sin barbijo | total |
| Observacion | Con barbijo | 9 | 1 | 10 |
| | sin barbijo | 1 | 9 | 10 |
| | total | 10 | 10 | 20 |

| | |
|---|---|
| Aciertos | 17 |
| Desaciertos | 3 |
| Exactitud | 0,9 |
| Taza de error | 0,1 |
| sencibilidad | 0,9 |
| especifidad | 0,9 |

| valores verdaderos | |
|---|---|
| | cantidad |
| sin barbijo | 10 |
| con barbijo | 10 |

Como podemos observar, se uso 20 datos para la evaluación, y se obtuvo un 90% de exactitud y un 10% de error,una sensibilidad de 90% y una especificidad de 90%, no está mal, pero si se aumentan los datos nos dará una mejor precisión y así reduciría la tasa de error ya que en la evaluación tiene un 98% de precisión y 2% de pérdida:



**CONCLUSIÓN**

Para concluir podemos decir que el uso de redes neuronales convolucionales nos ayudó a entrenar nuestro modelo y reducir los errores, con el uso de las 11 capas y sus filtros,también la capa de dropout fue importante, ya que eso nos ayudó a minimizar el sobreajuste de nuestro modelo, así como elegir el tamaño de épocas,ya que con más de 10 épocas igual presentaba un sobre ajuste.

**Bibliografia.-**

https://rpubs.com/chzelada/275494
https://www.incentro.com/es-ES/blog/que-es-tensorflow
https://www.kaggle.com/datasets/dsilverus/basededatosmie820