Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Momento de Retroalimentación Individual: Implementación de un modelo de Deep Learning.

Profesores:

Ivan Mauricio Amaya Contreras

Blanca Rosa Ruiz Hernandez

Antonio Carlos Bento

Frumencio Olivas Alvarez

Hugo Terashima Marín

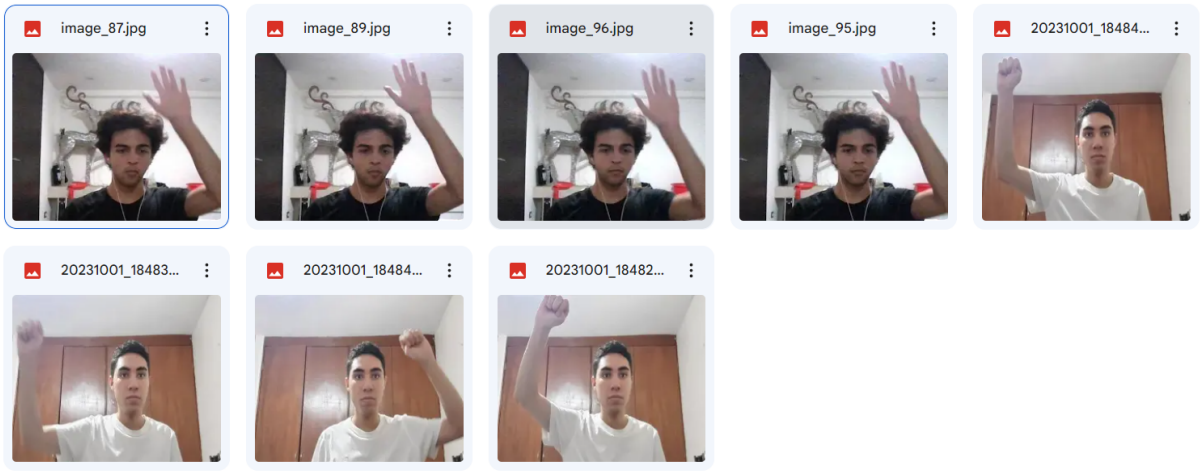
Julian Lawrence Gil Soares - A00832272

2 de Noviembre de 2023

Liga al CNN:
Momento de Retroalimentación Individual: Implementación de un mo...
Liga a datos de entrenamiento:
https://drive.google.com/drive/folders/1ekI5ggYAJD56_wb8uSUOqBQDqC0kqj2H?usp=sharing
Liga a datos de validación:
https://drive.google.com/drive/folders/1wVKbnvLIaurGT4YT-BohBl3VEak_Oq5h?usp=sharing
Liga a datos de prueba:
https://drive.google.com/drive/folders/1QTl70pZBEe6Ql2_psw5AEi6Ei-A0kP1g?usp=sharing

Las convolutional neural networks o CNN son redes neuronales que utilizan capas específicas para ser más eficientes en su reconocimiento y clasificación de imágenes, videos y audios. Existen varias capas dentro de una CNN, juntas las capas distinguen features específicas dentro del grid de la imagen. Una de las razones principales por las que las CNN son tan eficientes para la distinción de imágenes es porque cuentan con aprendizaje jerárquico. El aprendizaje jerárquico quiere decir que se transmiten features a través de las capas para poder mejor encontrar features.

Para mi proyecto personal decidí relacionarlo al reto así que cree un CNN que detecta si hay una mano presente en una imagen o no. El primer paso para esto fue generar un dataset de entrenamiento, validación y prueba con imágenes tanto positivas como negativas. Utilice imágenes que tomé de mi y otro integrante de mi equipo para esto.



En total conseguí 100 imágenes negativas y 169 negativas. Después distribuir las imágenes entre el dataset de entrenamiento, validación y

prueba. Una vez que obtuve mi dataset empecé a programar mi CNN. Para su implementación utilice Tensor Flow y Keras. El primer paso fue cargar mis datos de entrenamiento y validación. Utilizo un modo de clasificación binaria ya que el resultado que quiero es clasificar si la mano está presente o no está presente en la imagen.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

image_height = 640
image_width = 480
batch_size = 32

train_data_gen = ImageDataGenerator(
    rescale=1./255,
)

#training
train_generator = train_data_gen.flow_from_directory(
    '/content/drive/MyDrive/training',
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='binary',
    subset='training'
)

#validation
validation_generator = train_data_gen.flow_from_directory(
    '/content/drive/MyDrive/Validation',
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
)
```

Después envía esos datos a mi modelo para entrenarlo y para revisar las métricas de desempeño de mi modelo, yo utilizo el valor del accuracy para determinar que tan buen desempeño tiene mi modelo. Mi modelo tiene un accuracy de 1.0 así que le atina al 100% de las clasificaciones que le envió.

```python
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Input(shape=(image_height, image_width, 3)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(train_generator, epochs=10, validation_data=valida

test_data_gen = ImageDataGenerator(rescale=1./255)

test_generator = test_data_gen.flow_from_directory(
    '/content/drive/MyDrive/testing',
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='binary'
)

test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc}')
```