Anti-Spoofing

Adrian Villalobos, Isaac Ramirez

10/05/2024

Anti-Spoofing

Eleccion de los modelos

Antes de elegir el modelo de yolo, se hizo una busqueda al respecto de cuales son los mejores modelos para la deteccion deimagenes, se realizaron comparaciones entre Yolo y VGG. Pero se concluyo que Yolo era mejor para el contexto necesario debido a que VGG es mucho mas pesado tanto eficientemente como en tiempo de ejecucion. Por otro lado se analizaron otras librerias de deep-learning como TensorFlwo y PyTorch, pero se concluyo que Yolo era la mejor opcion para el proyecto.

Yolo

Yolo es un modelo de deep-learning que tiene caracteristica sque lo destacan de los demas, como por ejemplo el ReLu, que permite que el modelo obtenga una mejor precision en la deteccion de objetos. Tiene dimensiones de entrada mas grandes, esto con el fin de poder pasar imagenes con mayor calidad, para que el algoritmo pueda detectarlas de mejor manera.

Primero se analisis de la problematica con el fin de indentificar cual modelo era el mas adecuado, por lo tanto se llego a la conclusion de que se va a utilizar el modelo clasificador(Classifier) de Yolo.

Creacion del Dataset

Primero se les pidio ayuda a conocidos y a personas dispuestas a realizar un video de alrededor de 1 minuto, perfilando la cara por la mayoria de angulos posibles, tambien realizando distintos tipos de expresiones. Esto con el fin de tener un grupo de imagenes para el dataset. Por otro lado se buscaron imagenes en internet y luego con esas imagenes obtenidas se procesaron con el fin de obtener las imagenes con un estilo de spoofing. Esto se realizo con distintos tipos de camaras y distintos tipos de paneles en las pantallas para que luego cuando se creara el dataset el modelo tuviera distintos tipos de imagenes con diferentes contextos para analizar.

El dataset obtenido cuenta con 6000 imagenes, clasificadas en 80% (4800 imagenes aprox), 10% (600 imagenes aprox), 10% (600 imagenes aprox).

Entrenamiento

El entrenamiento se realizo en python, con la libreria de Ultralytics, luego se entreno el modelo por primera vez se entreno con 50 epochs, luego 100 epochs y por ultimo 130 epochs, pero se concluyo que al entrenar tanto el modelo pierde su consistencia y credibilidad. Por lo tanto se dejo con 100 epochs.

Se eligio el modelo M(Medium), debido a su tamaño, eficiencia y credibilidad, ya que con el modelo XL, el modelo no servía de la manera correcta y daba analisis erroneos.

Una vez se eligió el modelo, se comenzó con el entrenamiento

```
# Comando para comenzar el entrenamiento

# imgsz = Image size of the input
# batch = Indicates how many images are processed
#before the model's internal parameters are updated.
# weight_decay = Penalizing large weights
#to prevent overfitting.
```

```
model.train(data=dataset, epochs=100, imgsz=244, batch=16, weight_decay=0.0005)
```

Luego se utilizó el `model.val`, con el fin de que el modelo use un 10% de los datos para cambiar hiperparametros y mejorar el modelo. Al final de el documento se encuentra la imagen de la confusion-matrix obtenida mediante la evaluación del modelo.

Implementación

Una vez se obtuvo el modelo entrenado, se comenzó a relizar un algoritmo capaz de detectar los rostros y luego pasar el ROI(Region of Intrest) hacia el modelo de Yolo para su analisis, en esta etapa hubo un problema y fue que se estba utilizando el modelo predeterminado sin entrenar de CV2, que es HaarCascade, pero se concluyó que no era el más apto debido a su credibilidad, ya que en la mayoría de ocaciones no detectaba los rostros de manera correcta. Por lo tanto se elegió el modelo de Yunet, debido a su alta credibilidad y eficiencia detectando rostros, en distintos contextos. Una vez se detecta el rostro y se envia su ROI a el algoritmo de yolo, el analiza la imagen y así da una estimación de la credibilidad de cada categoría, si es mayor a 0.75 se considera que es un rostro real, sino se considera que es un rostro falso.

Blink Detector

Otra problematica a la que se enfrentan este tipo de algoritmos son las imagenes muy realistas o en muy buena calidad debido a que los algoritmos se les dificulta encontrar diferencias en este tipo de imagenes, por lo tanto acá es dónde se implementan este tipo de medidas, ya que el algoritmo de Blink Detection lo que hace es detectar si una persona parpadea o no, si no parpadea se descarta la imagen, si parpadea se considera que es un rostro real.

Al ser como tal un procedimiento no se necesitaba un dataset. Se utilizó un modelo de dlib, llamado shape_predictor_68_face_landmarks, que es un modelo pre-entrenado que detecta todos los rasgos faciales y los puntos de los ojos, con el fin luego de hacer el analisis de los puntos de los ojos para ver si parpadea o no.

3D Face Model

Este modelo fue entrenado con distintas imagenes de rostros generados por computadora para el caso del Spoof y por rostros utilizados en el dataset del primer modelo para el caso del Live. Es uno de los modelos que mejor funciona gracias a su baja complejidad, ya que, este proceso de clasificacion es notoriamente mas sencillo que los procesos mencionados anteriormente.

Por otro lado, como ya se menciono la baja complejidad del problema, se opto por un clasificador de yolo de menor nivel, se podria decir una version mas “Lite”, esto para profundizar en la rapidez del modelo

Google Colab

Por último se implementaron los 3 algoritmos en Google Colab, con el fin de que sea mucho más sencillo para las personas probar este tipo de algoritmos. Se recomienda usar los algoritmos localmente con el fin de tener más rendimiento y eficiencia a la hora de usarlos. *Documento de Google Colab*

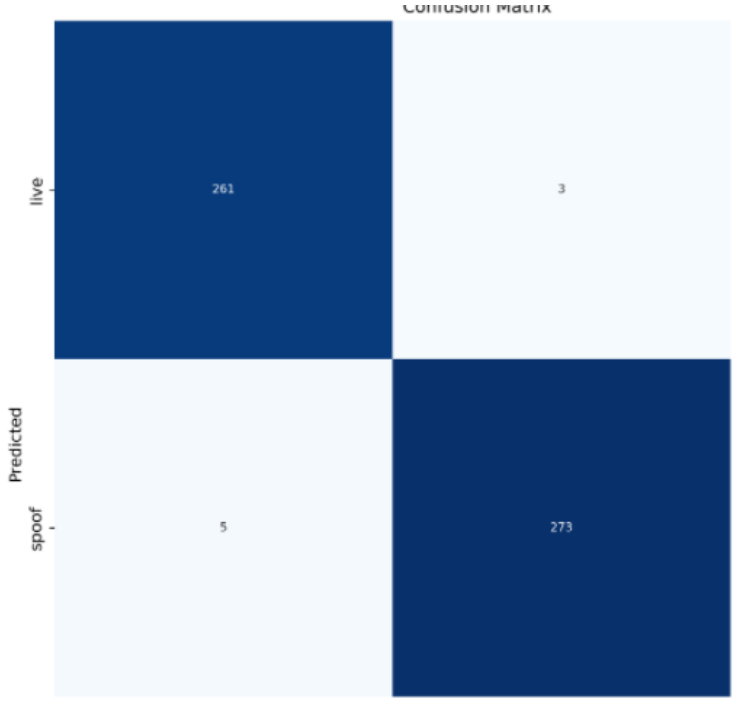

Figura 1: Confusion Matrix obtenida del algoritmo de Yolo para la Calidad de Imagen

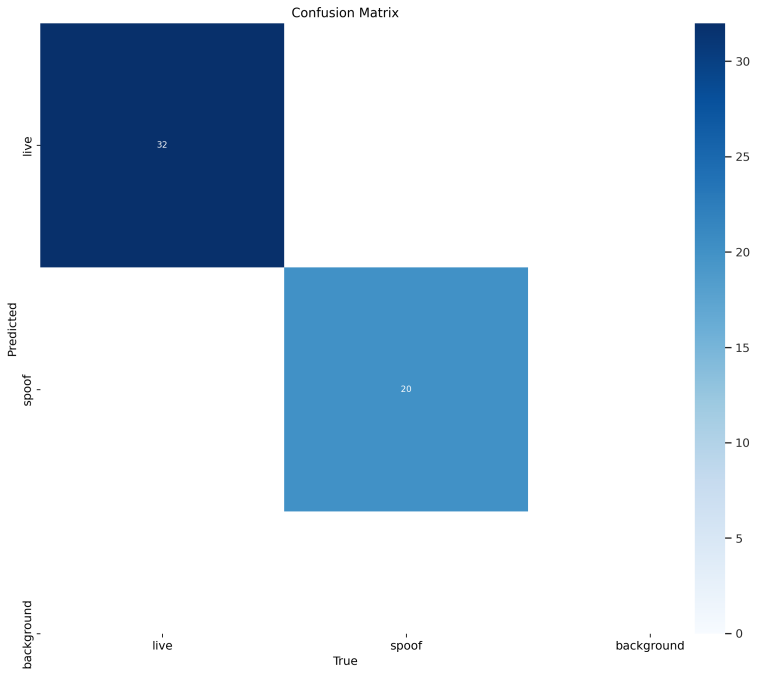

Figura 2: confusion Matrix del Algoritmo de 3D Face Model