

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Modelo detección de intrusos para prevenir robos | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Lucas Porras Y Santiago Avilés |  | | | |
|  | | | | 28/03/2025—Ciencia de datos—Alfredo Díaz |  | | | |
|  | | |  | | |  | | |

# Descripción del Proyecto

Este proyecto implementa un sistema de detección de intrusos en tiempo real utilizando inteligencia artificial. Se considera intruso a cualquier persona que lleve una gorra, un tapabocas o un casco de motocicleta. La solución emplea el modelo MTCNN para la detección de rostros y una red neuronal basada en transferencia de aprendizaje (MobileNetV2) para la clasificación de los rostros detectados. Cuando se identifica un intruso, el sistema emite una alerta sonora y muestra un mensaje en pantalla.

# Metodología

El desarrollo del proyecto sigue un enfoque estructurado en varias fases, cada una de ellas diseñada para garantizar la precisión y eficiencia del sistema de detección en tiempo real. A continuación, se detallan las fases del proyecto:

La primera fase consiste en la **configuración del entorno**, donde se instalan todas las herramientas y librerías necesarias para el desarrollo del proyecto. Se utiliza Google Colab como entorno de desarrollo debido a su capacidad para ejecutar código en la nube y aprovechar GPUs de alto rendimiento sin necesidad de configuraciones complejas. Se instalan TensorFlow, PyTorch y OpenCV, bibliotecas esenciales para el procesamiento de imágenes y la implementación de modelos de aprendizaje profundo. Google Colab permite el uso de GPU acelerada mediante la configuración de CUDA, lo que mejora significativamente el rendimiento del modelo sin necesidad de hardware local especializado. Además, se suben los datos al entorno de Colab y se gestionan mediante Google Drive para facilitar el acceso y procesamiento.

En la segunda fase, se realiza el **preprocesamiento de datos**, lo que incluye la recopilación y organización de imágenes. Las imágenes de entrenamiento están clasificadas en carpetas que representan diferentes categorías: personas con gorra, personas con tapabocas, personas con casco y personas sin ningún accesorio. Para mejorar la precisión del modelo, se utiliza el detector de rostros MTCNN, que permite extraer únicamente los rostros de las imágenes y descartar el fondo, lo cual reduce el ruido en los datos. Estos rostros detectados se almacenan en una nueva estructura de carpetas organizadas, facilitando el proceso de entrenamiento del modelo. También se aplica un proceso de aumento de datos, que incluye técnicas como rotación, escalado y espejado de imágenes, con el objetivo de mejorar la generalización del modelo y evitar el sobreajuste.

# Configuración del Entorno

Para llevar a cabo este proyecto, es fundamental establecer correctamente el entorno de desarrollo. Se opta por Google Colab, lo que permite acceder a recursos de cómputo avanzados sin requerir hardware especializado. Se instalan las bibliotecas esenciales como TensorFlow, PyTorch y OpenCV, asegurando que sean compatibles con la GPU proporcionada por Google Colab.

# Preprocesamiento de Imágenes

Luego, se realiza la estructuración de los datos, donde se recopilan imágenes y se organizan en carpetas representando cada una de las categorías (gorra, tapabocas, casco). Para mejorar la calidad del conjunto de datos, se utiliza MTCNN, que detecta los rostros en las imágenes y los recorta, eliminando cualquier información irrelevante. Además, se implementa un aumento de datos con transformaciones como rotación y espejado, con el fin de aumentar la diversidad del conjunto de entrenamiento y mejorar la generalización del modelo.

# Entrenamiento del Modelo

Posteriormente, se entrena una red neuronal convolucional basada en MobileNetV2, un modelo preentrenado en ImageNet que permite una clasificación eficiente. Se eliminan sus capas superiores y se agregan capas específicas para nuestro caso de uso, aplicando técnicas de ajuste fino para optimizar su rendimiento. El entrenamiento se lleva a cabo con un conjunto de datos balanceado, ajustando hiperparámetros como la tasa de aprendizaje y el número de épocas para maximizar la precisión del modelo. Una vez entrenado y validado el modelo, se guarda para su implementación en tiempo real.

# Detección en Tiempo Real

Se captura video en vivo con OpenCV y se detectan rostros usando MTCNN. Cada rostro detectado se procesa y se pasa al modelo entrenado para su clasificación. Si se identifica un accesorio considerado intrusivo, se emite una alerta sonora y se muestra un mensaje en pantalla. Además, se registra la detección en un archivo de log para su análisis posterior.