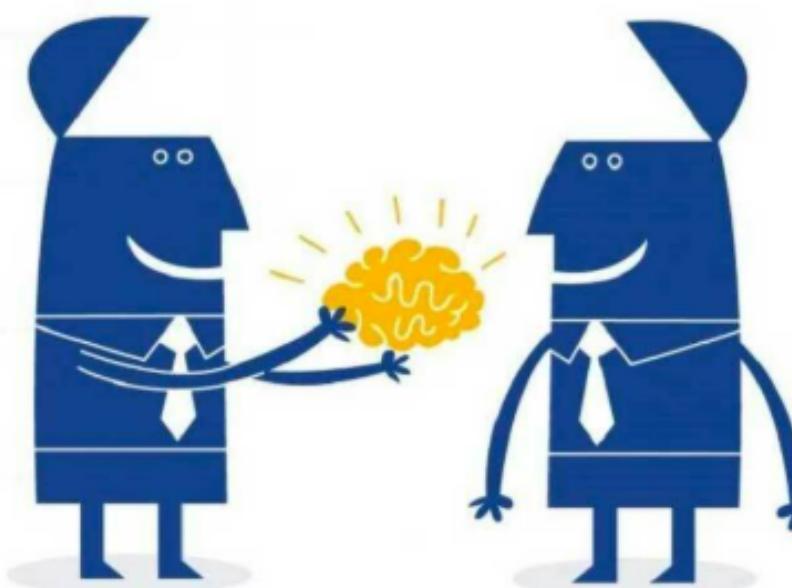


Transfer Learning orientado al edge computing: detección de EPIs en Raspberry Pi



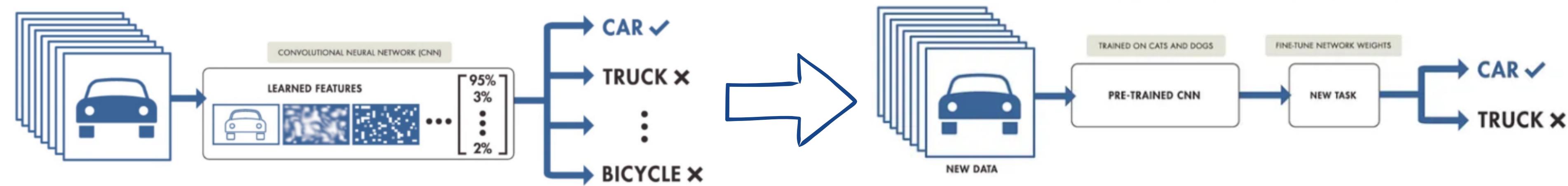
Manuel Fernández Giráldez
Nuria Sorrentino Rebull

Índice

1. ¿Qué es el Transfer Learning?
2. Descripción del problema
3. Descripción del dataset
4. Metodología y entrenamiento del modelo
5. Resultados del entrenamiento
6. Despliegue en Raspberry Pi
7. DEMO
8. Conclusiones y trabajos futuros
9. Bibliografía

¿Qué es el Transfer Learning?

Técnica de aprendizaje automático dónde se utiliza un modelo ya entrenado para una tarea específica, se ajustan algunas capas o se entrena una parte pequeña del modelo para adaptarlo a un nuevo problema con menos datos y menos tiempo de cómputo.



- ✓ Menos datos: El modelo ya viene “pre-aprendido”, así que no se necesita una base de datos enorme.
- ✓ Menor coste computacional y menor tiempo de entrenamiento: Entrenas solo unas capas finales y buenos resultados en mucho menos tiempo que entrenando desde cero
- ✓ Mejor rendimiento con pocos datos: Suele dar mejor precisión cuando el dataset es pequeño o moderado.
- ✓ Reutilización de modelos SOTA: Aprovechar modelos ya entrenados por grandes empresas o comunidades (ResNet, EfficientNet, etc.).

Descripción del problema



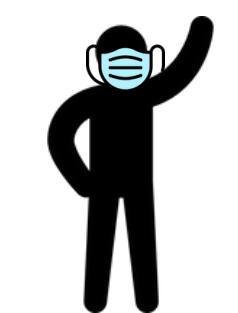
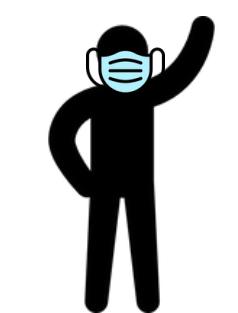
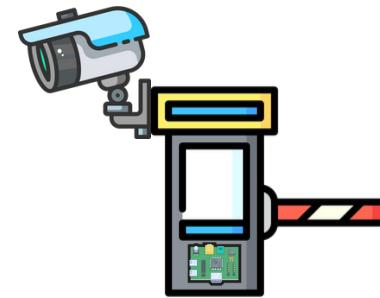
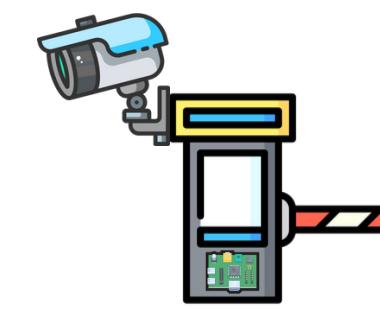
Automation in Construction
Volume 171, March 2025, 105987



Detection of helmet use among construction workers via helmet-head region matching and state tracking

Yi Zhang ^b✉, Shize Huang ^a✉, Jinzhe Qin ^b✉, Xingying Li ^b✉, Zhaoxin Zhang ^c✉,
Qianhui Fan ^b✉, Qunyao Tan ^b✉

- Monitorizar el uso de EPIs en entornos laborales hostiles
- Se propone una solución que funciona como sistema de acceso a entornos en los que sea necesario estar usando mascarilla o casco de seguridad



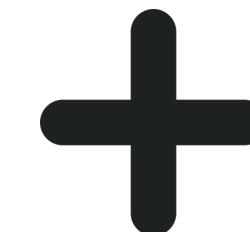
Descripción del dataset



Face Mask Detection ~12K Images Dataset

12K Images divided in training testing and validation directories.

[kaggle.com](https://www.kaggle.com)



helmet Classification Dataset by Classification

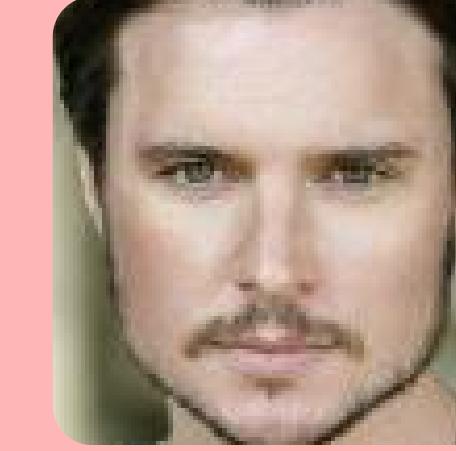
2886 open source waring images. helmet dataset by Classification

[Roboflow](#)

Clase 1: MASCARILLA



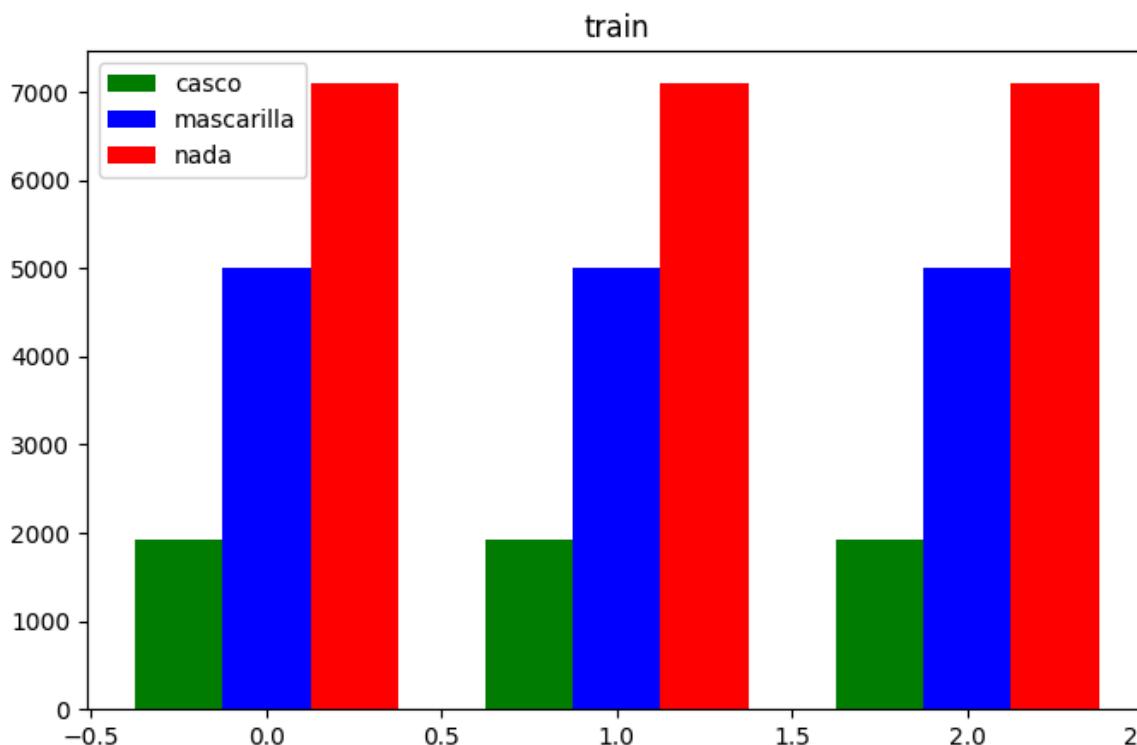
Clase 2: CASCO



Clase 3: NADA

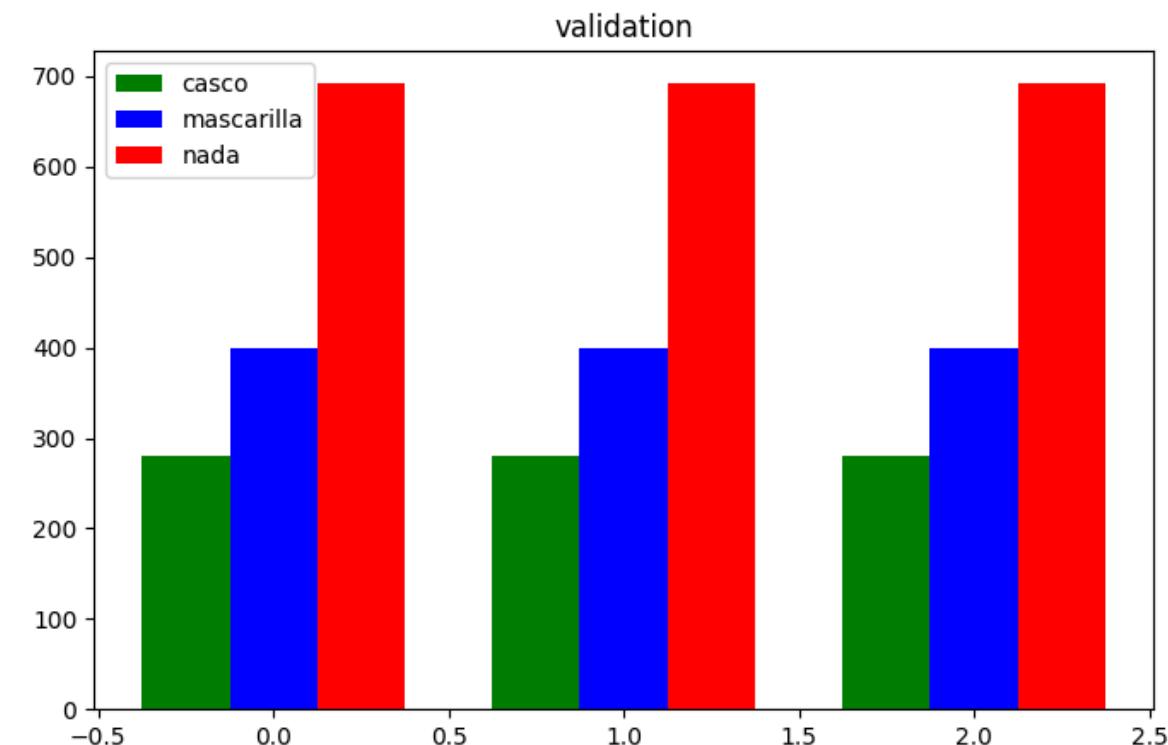
Descripción del dataset

- Antes de balanceo de clases:



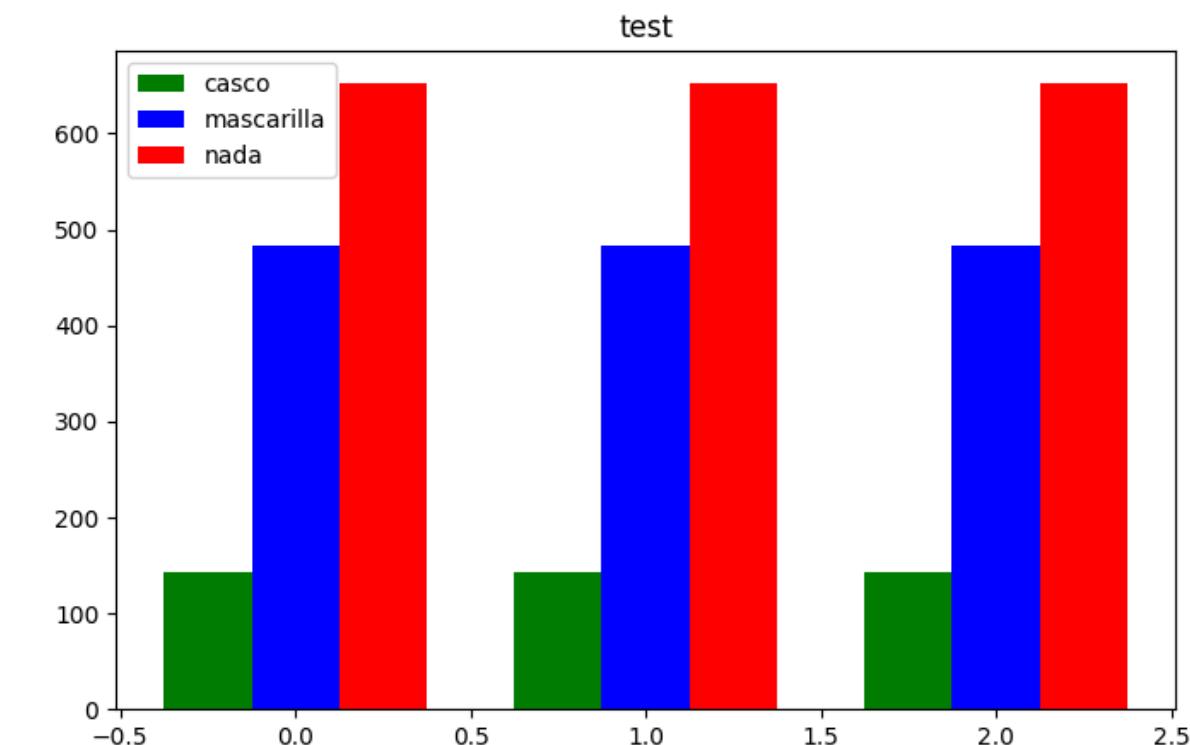
Train

casco: 1926
mascarilla: 5000
nada: 7106
Total: 14032



Validation

casco: 281
mascarilla: 400
nada: 693
Total: 1374

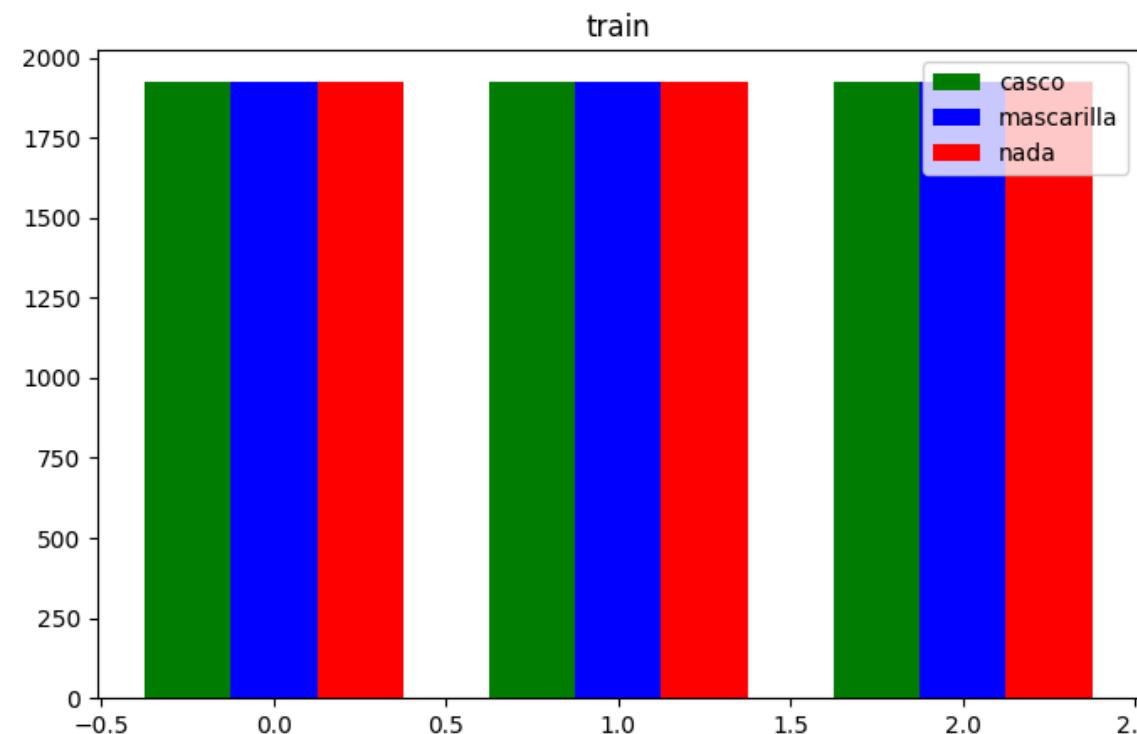


Test

casco: 143
mascarilla: 483
nada: 653
Total: 1279

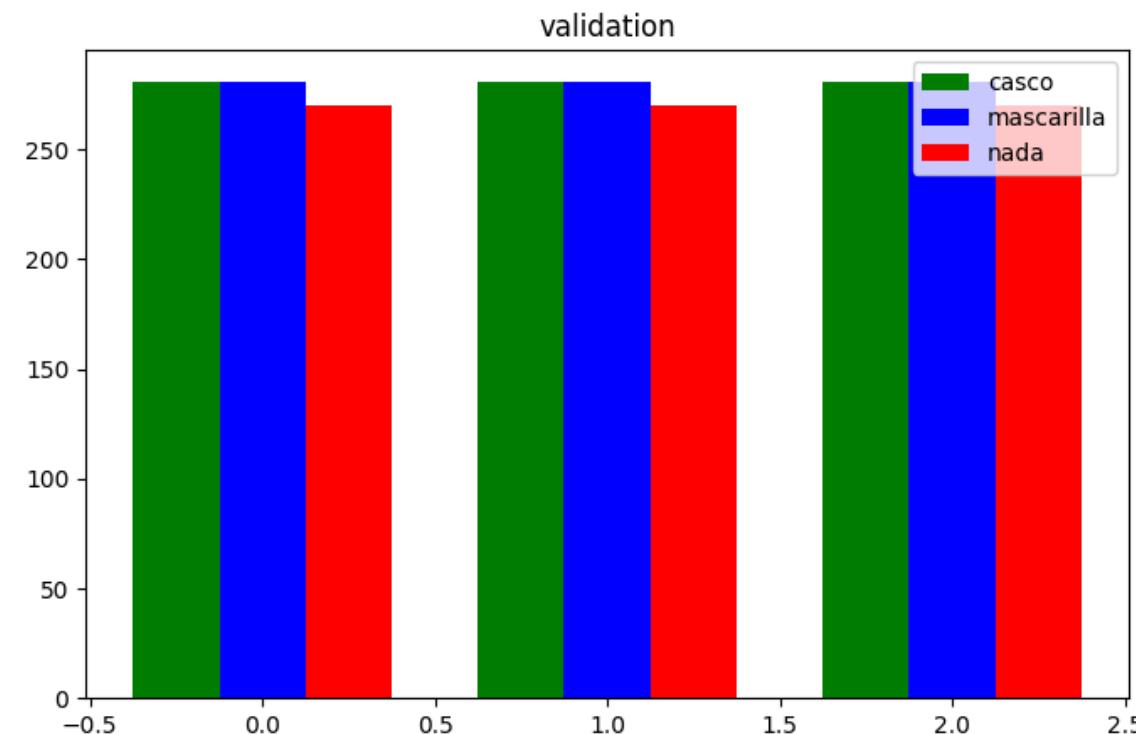
Descripción del dataset

- Después de balanceo de clases:



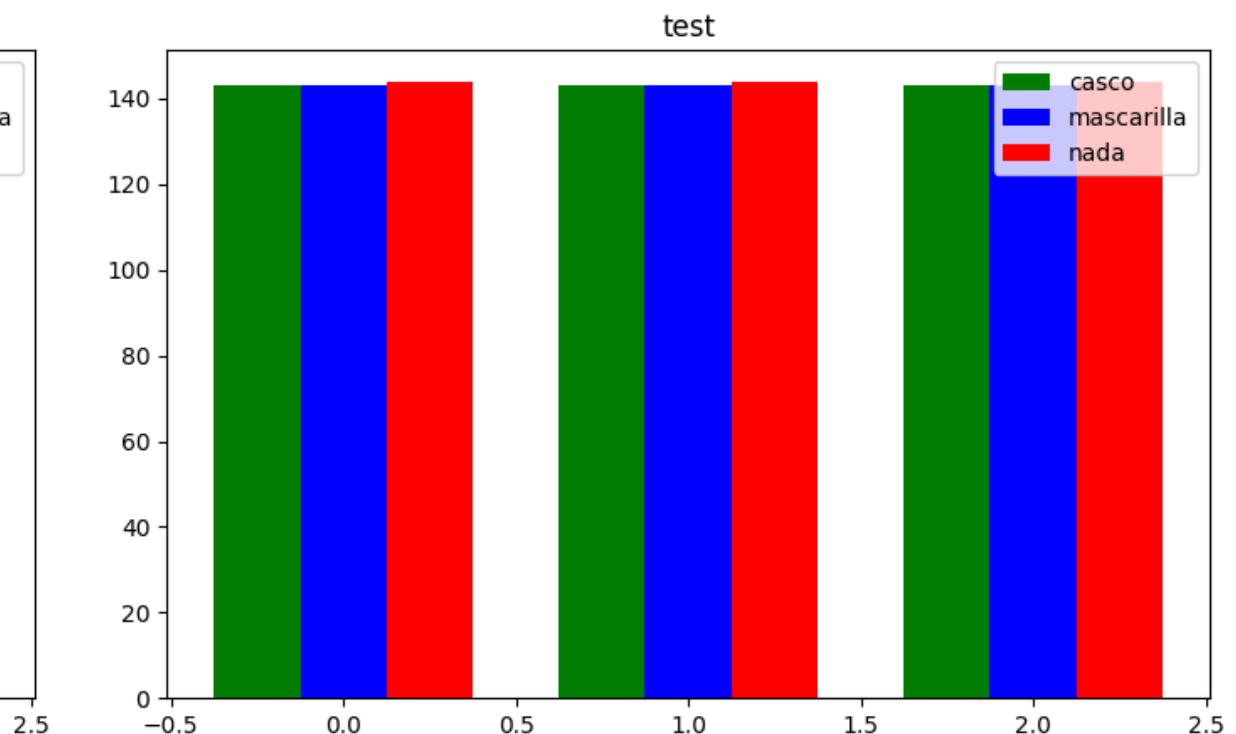
Train

casco: 1926
mascarilla: 1926
nada: 1926
Total: 5778



Validation

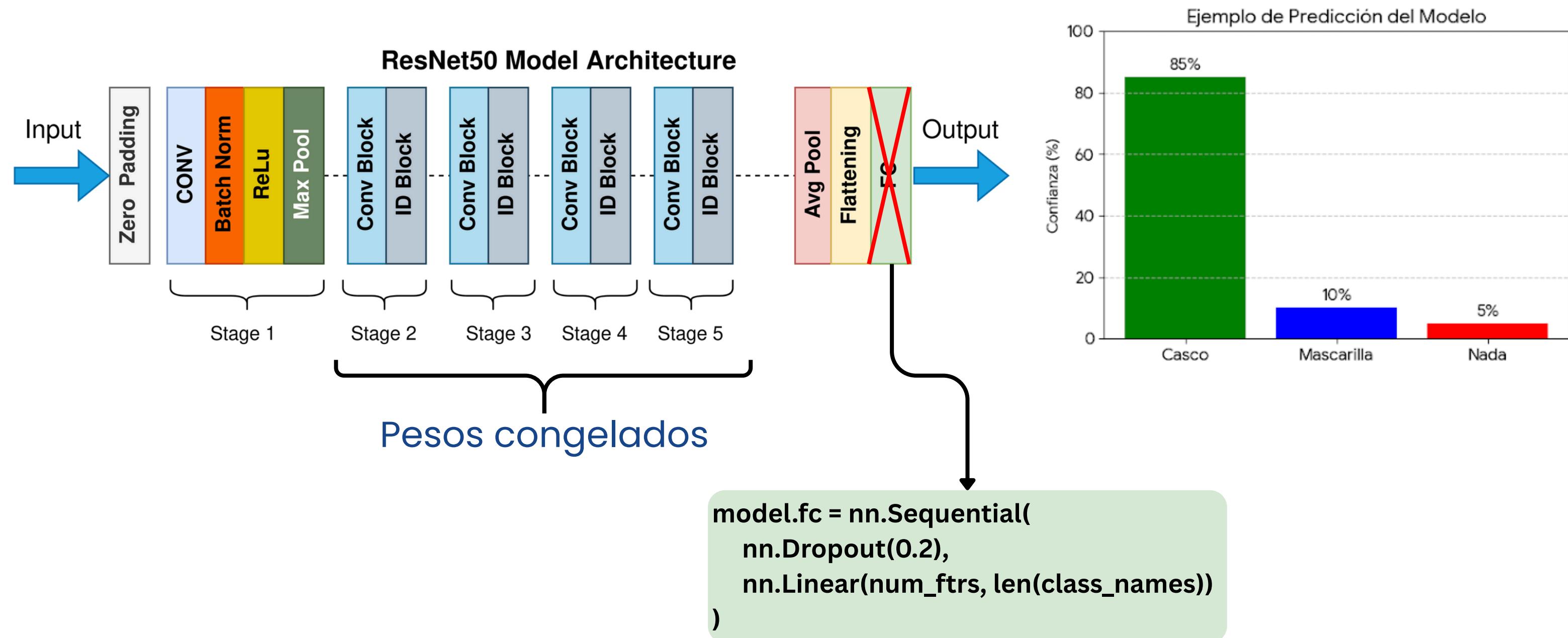
casco: 281
mascarilla: 281
nada: 270
Total: 832



Test

casco: 143
mascarilla: 143
nada: 143
Total: 429

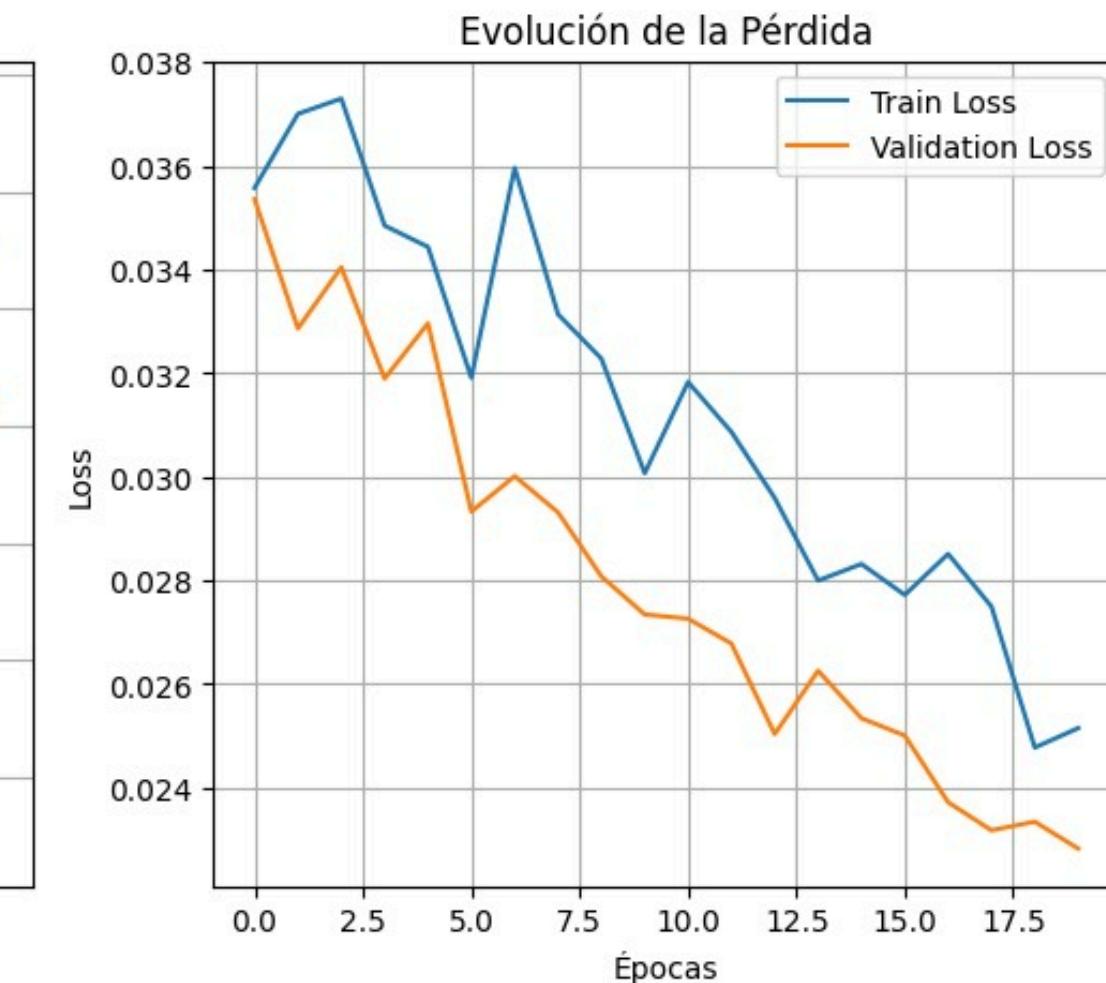
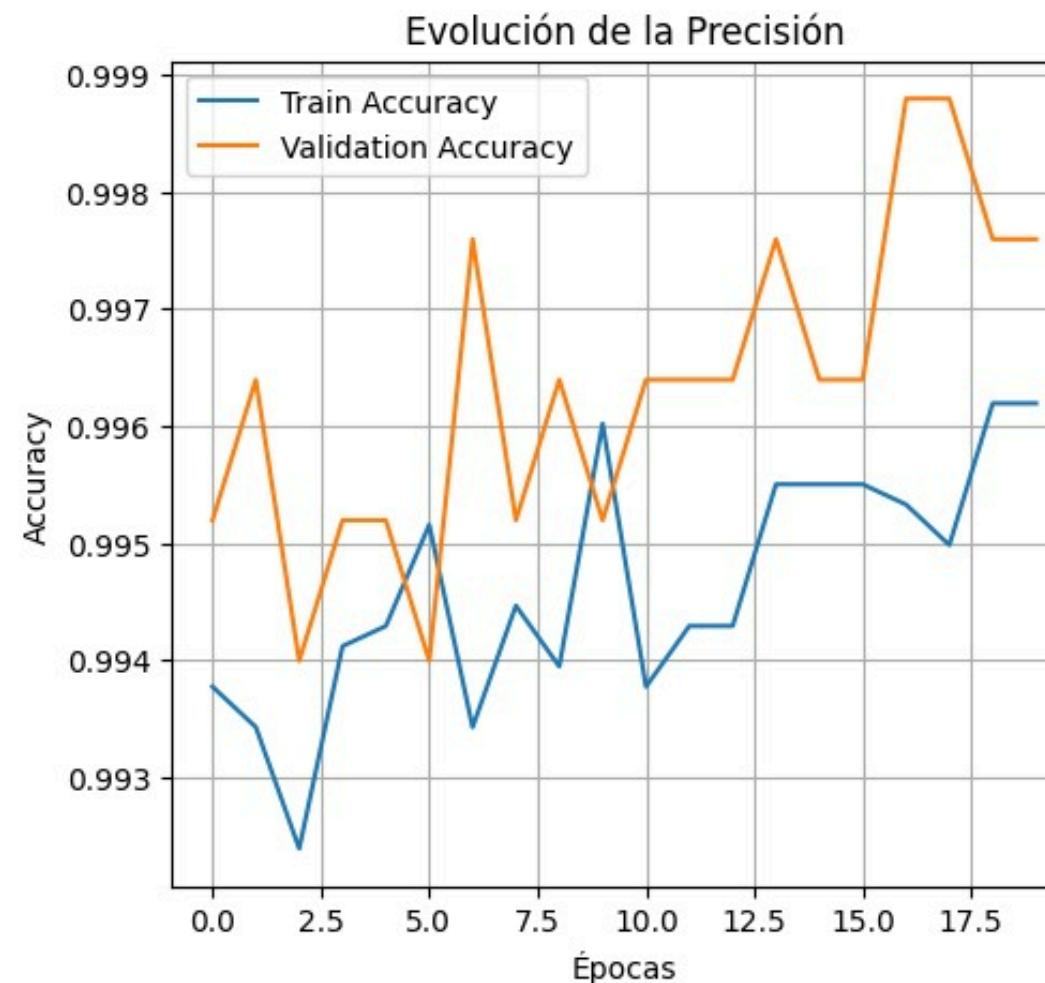
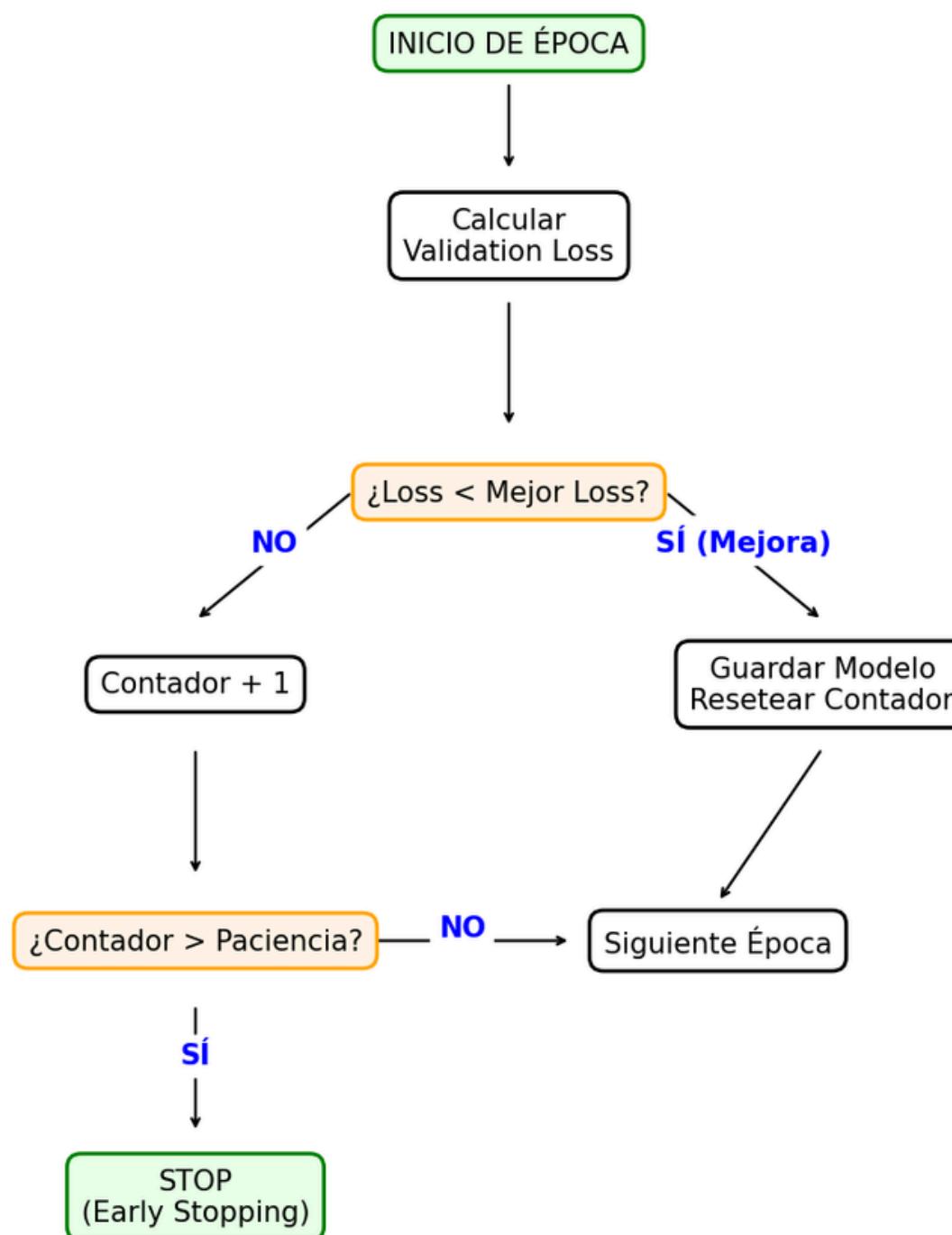
Metodología y entrenamiento del modelo



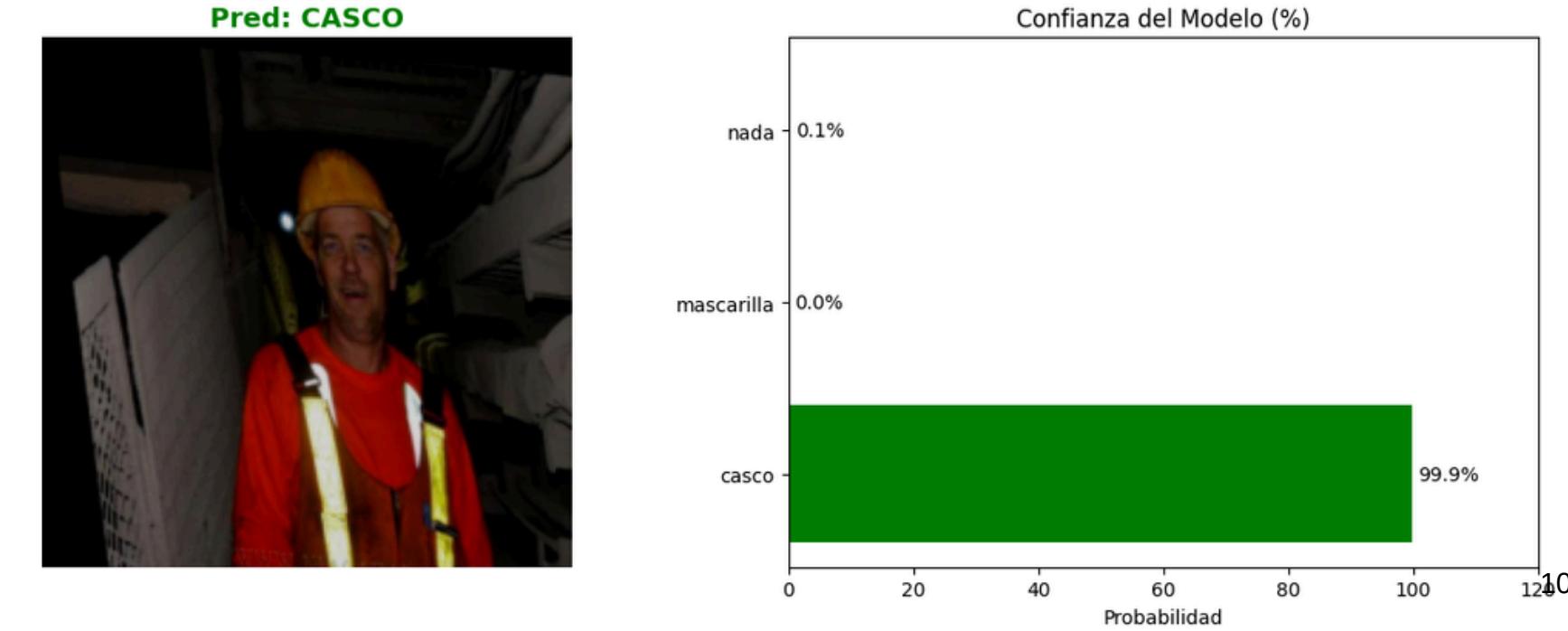
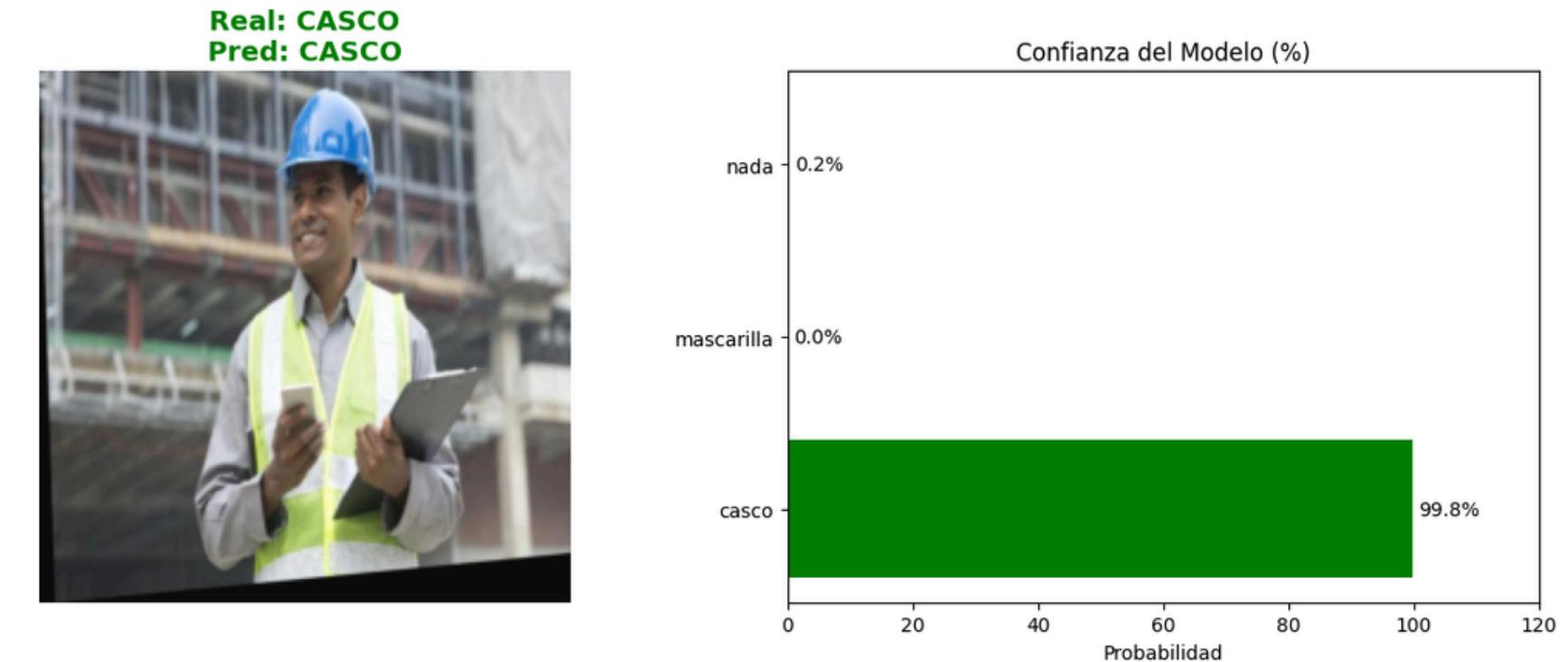
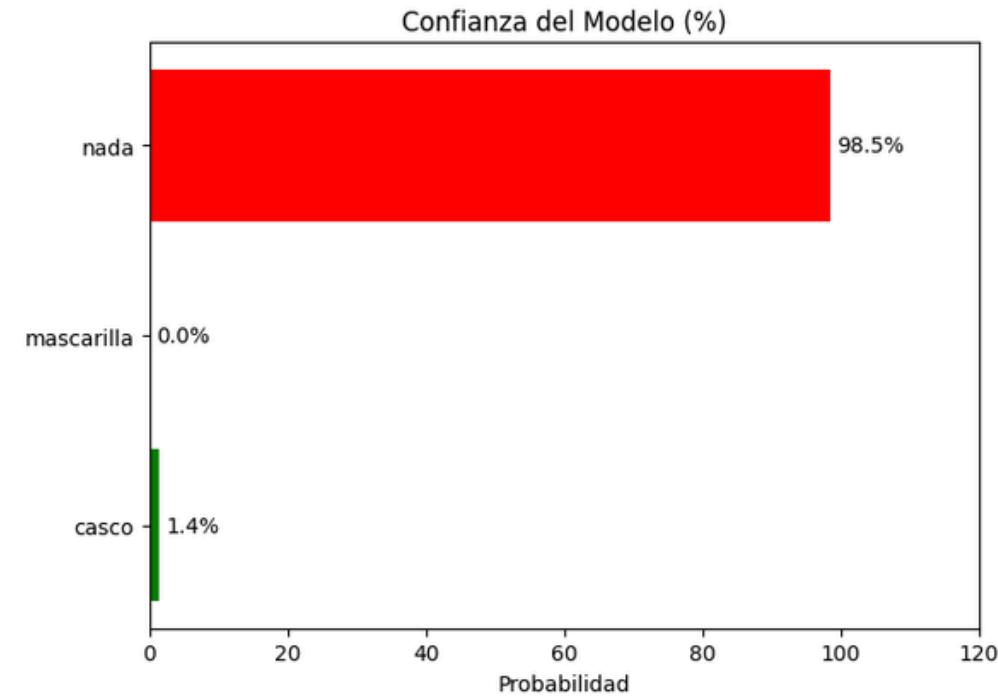
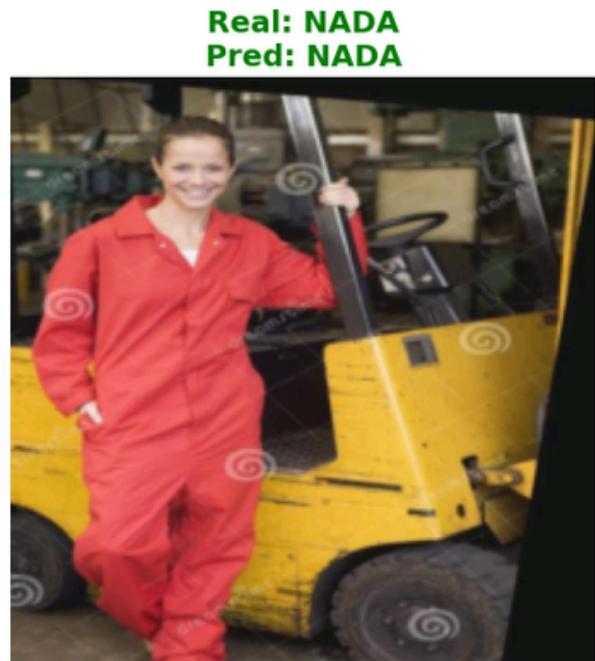
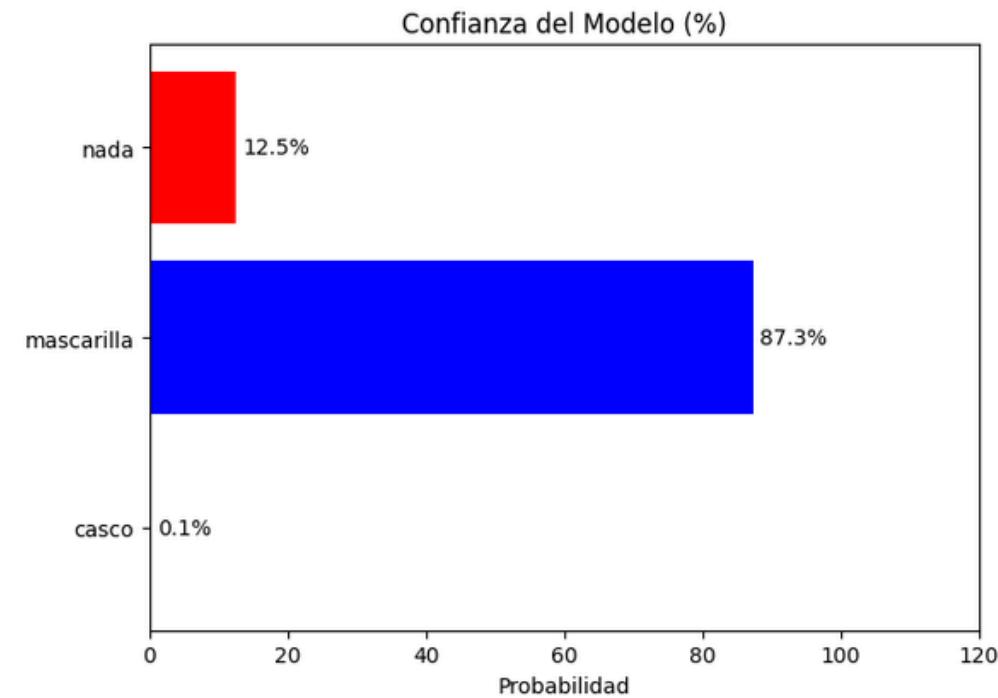
Metodología y entrenamiento del modelo

Funcionalidad adicional: Early Stopping

Lógica de Early Stopping



Resultados de entrenamiento



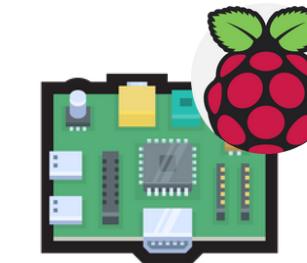
Despliegue en Raspberry Pi

Configuración inicial

- Se crea un único fichero llamado **inferencia.py**
- Se definen unas variables de configuración:
 - Ruta con la configuración del modelo a cargar en formato **.pt**
 - Frecuencia de procesado de frames
 - Pipeline de procesado de las imágenes

Se crean 2 funciones:

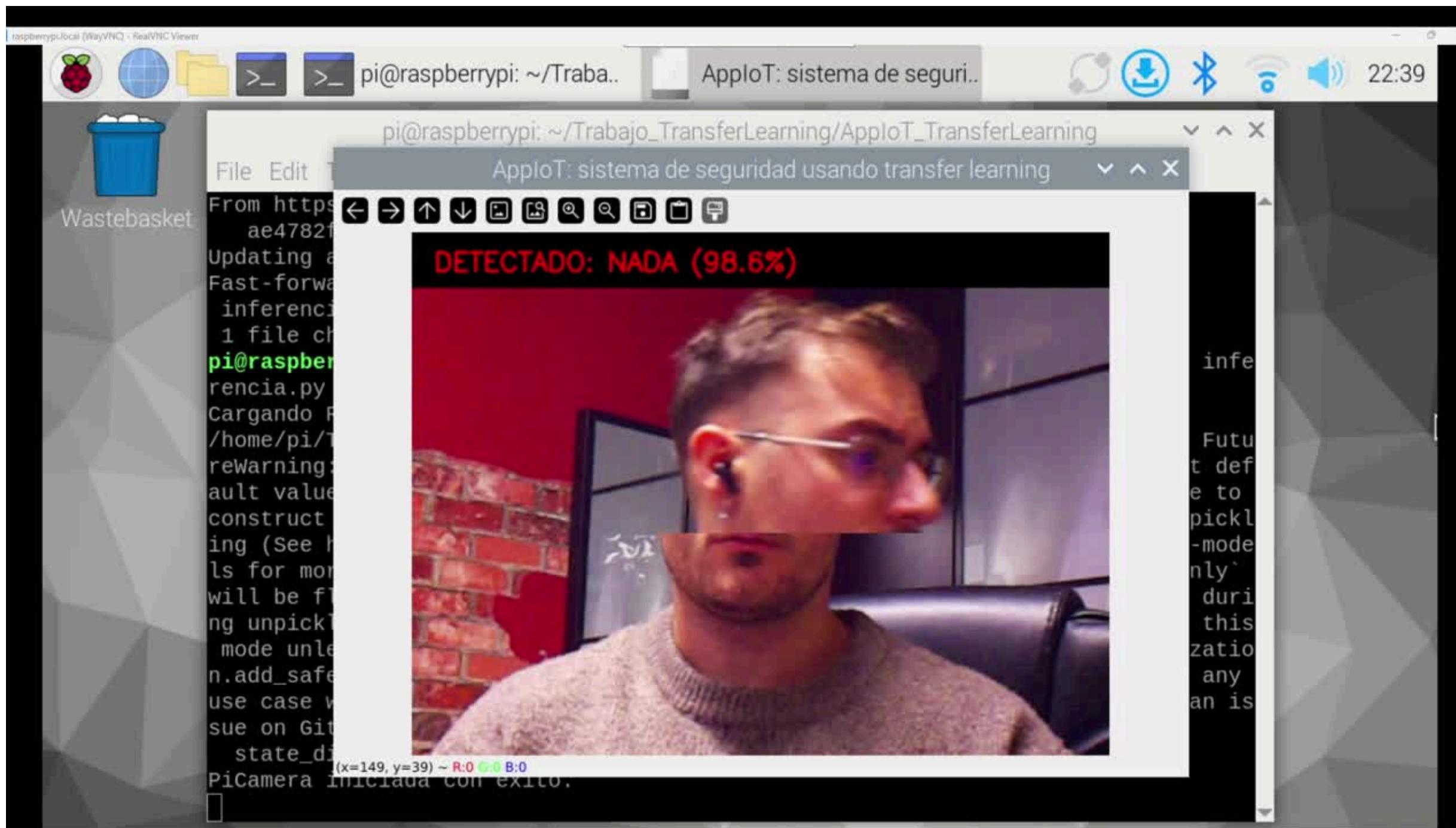
- Función **load_model()**:
 - Crea el modelo ResNet50, le añade la cabeza con la ANN FC 2048 x 3, carga los pesos entrenados y pone el modelo en modo evaluación
- Función **predict_frame(model, frame_rgb)**:
 - Pone la imagen en formato array, añade el tamaño del batch con **unsqueeze(0)**
 - Le pasa la imagen al modelo y calcula las probabilidades



Bucle principal

1. Se llama a la función **load_model()**
2. Captura una imagen con la función **capture_array**, que entrega un array con **[alto, ancho, 3 canales]**. Se cargan directamente en RAM y en formato BGR
3. Si ese frame toca evaluación (1 de cada 4), se llama a la función **predict_frame**
4. Se imprime por pantalla la clase detectada y su probabilidad

DEMO



Conclusiones y trabajo futuro

✓ El transfer learning con ResNet50 ha permitido **obtener un modelo de detección de EPIs** (casco, mascarilla, nada) con **buenas precisiones sin necesidad de grandes datasets ni hardware potente**.

✓ El **preprocesado del dataset (balanceo de clases)** y el uso de **early stopping** han sido clave para **evitar sesgos y sobreajuste**, logrando un **modelo que generaliza bien**.

✓ El **despliegue en Raspberry Pi** demuestra que **es viable ejecutar la inferencia en el borde (edge computing)**, con baja latencia y sin enviar imágenes a la nube.

✓ La **solución es escalable**: se puede ampliar a **más tipos de EPIs, múltiples personas por imagen e integración con plataformas IoT de seguridad laboral**.

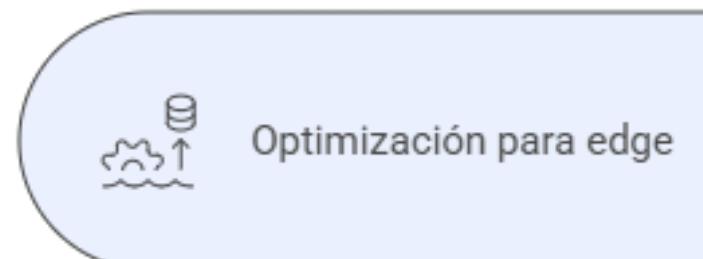


Detección de objetos

Detectar varias personas y EPIs en la misma imagen, usando modelos tipo YOLO/SSD.

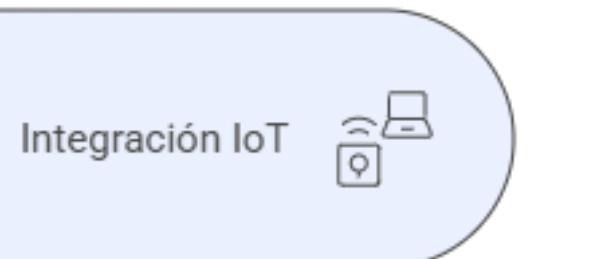


Mejorar el dataset



Optimización para edge

Probar modelos más ligeros, cuantización o poda de pesos para reducir tamaño y acelerar la inferencia.



Integración IoT

Enviar las detecciones a una plataforma IoT para generar alarmas, estadísticas y dashboards.

Bibliografía

- CNN – Transfer Learning.pdf, S.Toral.

Muchas gracias

¿Preguntas?