Teoría de Aprendizaje de Máquina - 2025-1

Guía para la Presentación del Proyecto Final de Curso

Profesor: Andrés Marino Álvarez Meza, PhD
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Universidad Nacional de Colombia - Sede Manizales
-Juan Esteban Villada Sierra
-Sergio Andrés Mora Orrego
-Mateo Duque Gaviria

Tema general

Aplicación de técnicas de *Machine Learning y Deep Learning* en tareas de interés.

Abstract

Este proyecto implementa un sistema de control de acceso basado en reconocimiento facial utilizando YOLOv8. Se entrenó un modelo personalizado con un dataset creado en Roboflow, se optimizó para ejecución en dispositivos embebidos (Jetson Nano) mediante TorchScript, y se desarrolló una interfaz gráfica en Python/Tkinter para validación local. El sistema alcanzó un 92 % de precisión en pruebas de laboratorio, demostrando la viabilidad de la solución en entornos restringidos computacionalmente.

1 Motivación del Proyecto

Los sistemas convencionales de control de acceso no pueden adaptarse fácilmente a condiciones cambiantes (como iluminación o ángulos) ni operar

eficientemente en hardware limitado. Se requiere una solución basada en visión por computadora e inteligencia artificial que funcione en tiempo real y con recursos restringidos.

El control de acceso tradicional mediante tarjetas o contraseñas presenta vulnerabilidades de seguridad. Este proyecto busca:

- Automatizar la verificación de identidad en espacios restringidos
- Reducir costos asociados a sistemas comerciales
- Explorar técnicas de Edge AI para procesamiento *on-device*

El uso de técnicas de machine learning y deep learning en este proyecto es clave para lograr un reconocimiento facial preciso y en tiempo real, algo que los métodos tradicionales no pueden ofrecer. Técnicamente, permite ejecutar modelos avanzados como YOLOv8 en dispositivos embebidos como Jetson Nano mediante TorchScript, optimizando rendimiento y eficiencia. Socialmente, mejora la seguridad y agiliza el acceso sin contacto. Económicamente, representa una solución de bajo costo al emplear herramientas de código abierto. Y científicamente, promueve la aplicación real de IA en entornos con recursos limitados, demostrando su viabilidad con un 92 % de precisión en pruebas controladas.

2 Planteamiento del Problema

¿Cómo implementar un sistema de control de acceso basado en reconocimiento facial que funcione en dispositivos embebidos de bajo costo, con un tiempo de inferencia menor a 500ms por frame?

Dominio:

- Dataset: 933 imágenes etiquetadas (2 clases: Acceso Permitido, Acceso Denegado.)
- Hardware objetivo: NVIDIA Jetson Nano (4GB RAM)
- Métrica principal: Precisión (Accuracy) >85 %

3 Estado del Arte

- **Reconocimiento facial utilizando redes neuronales artificiales en Raspberry Pi:** Este trabajo se centra en la creación de un prototipo de hardware de bajo costo para el reconocimiento facial.

  Los autores Propusieron un sistema biométrico de reconocimiento facial montado en una placa Raspberry Pi 3.

  Las pruebas demostraron que el sistema puede reconocer exitosamente a los sujetos con una tasa de precisión "bastante significativa y confiable". El mejor modelo entrenado alcanzó un

  Uno de los principales problemas que tuvieron para obtener resultados óptimos fue que el sistema depende de un ambiente bien iluminado, tanto para la captura de rostros para el entrenamiento como para el reconocimiento en tiempo real.

- **Access Control Using Facial Recognition with Neural Networks for Restricted Zones:**Este estudio se enfoca en la implementación de una solución robusta y escalable utilizando servicios en la nube para pequeñas y medianas empresas (PyMEs)

  Se enfocaron los autores en desarrollar un sistema de control de acceso para zonas restringidas en PyMEs, utilizando cámaras IP y redes neuronales.

  La tecnología demostró ser efectiva y precisa en la identificación de personas. Los resultados validan que la solución es eficiente y segura para el monitoreo y control de acceso.

  Para una implementación exitosa, es fundamental seleccionar cámaras IP de alta calidad y realizar un monitoreo y mantenimiento regulares del sistema.

- **Aplicación de algoritmos de redes neuronales y aprendizaje profundo para el reconocimiento facial en el control de asistencia: Una revisión de literatura:** Este documento no implementa un sistema, sino que analiza la investigación existente para identificar las técnicas más efectivas y los desafíos predominantes.

  El enfoque fue identificar qué algoritmos de deep learning y redes neuronales mejoran la precisión del reconocimiento facial para el control de asistencia e identificar los principales desafíos en su implementación

  Se fortalecio La revisión y concluyó que las Redes Neuronales Convolucionales (CNN) se utilizan ampliamente y son muy efectivas, con tasas

de precisión que oscilan entre el 95 % y el 99 %. Modelos comoVGGNet, ResNet y FaceNet son populares por su alta precisión

El procesamiento en tiempo real fue la limitancia a la necesidad de procesar grandes volúmenes de datos, esto requiere una infraestructura de hardware robusta, lo cual es una dificultad importante.

4 Objetivo del Proyecto

4.1 Objetivo General

Desarrollar e implementar un sistema de control de acceso mediante reconocimiento facial, basado en el modelo YOLOv10, que sea capaz de operar eficientemente en un dispositivo de computación de borde (NVIDIA Jetson Nano) con un tiempo de respuesta cercano al tiempo real.

4.2 Objetivos Específicos

- Adaptar y entrenar un modelo de detección de objetos (YOLOv10s) para la clasificación precisa de rostros en dos categorías: "Acceso Permitido" y "Acceso Denegado", utilizando un dataset personalizado.
- Optimizar el modelo entrenado mediante su conversión al formato TorchScript, con el fin de minimizar la latencia de inferencia y el consumo de recursos computacionales en el hardware limitado de la Jetson Nano.
- Validar cuantitativamente el rendimiento del sistema en términos de precisión y tiempo de inferencia, asegurando que cumple con los requisitos predefinidos (Precisión $> 85\,\%$, Tiempo de Inferencia < 500ms).
- Desarrollar una interfaz de usuario (GUI) en Python para la demostración, prueba y validación del sistema en un entorno local, capaz de procesar imágenes y mostrar los resultados de la detección.

5 Metodología Propuesta

La metodología se estructura en un pipeline de 5 etapas, desde la adquisición de datos hasta el despliegue final, garantizando la reproducibilidad y eficiencia del proyecto.

5.1 Adquisición y Configuración de Datos

Se utilizó la API de Roboflow para descargar una versión específica del dataset de rostros. Este enfoque asegura el versionado y la consistencia de los datos. Posteriormente, se creó programáticamente un archivo `data_config.yaml` que mapea las rutas de los conjuntos de *train*, *valid* y *test*, y define los nombres de las clases, un paso crítico para que el *framework* de ultralytics pueda localizar y entender los datos.

5.2 Preprocesamiento de Imágenes

Las imágenes son procesadas automáticamente por el *pipeline* de YOLO. Durante el entrenamiento (`imgsz=640`), las imágenes se redimensionan a 640x640 píxeles y se normalizan (los valores de los píxeles se dividen por 255.0) para adecuarse a la entrada de la red neuronal. Este mismo preprocesamiento se replica de forma manual en el *script* de inferencia para asegurar la consistencia.

5.3 Selección y Entrenamiento del Modelo

Se seleccionó el modelo YOLOv10s (`yolov10s.pt`) por su excelente balance entre velocidad y precisión, ideal para dispositivos de borde. Se realizó un proceso de *fine-tuning* sobre los pesos pre-entrenados en el dataset COCO. El entrenamiento se configuró para ejecutarse durante 100 épocas con un tamaño de lote (*batch*) de 16. El resultado de esta fase es el archivo de pesos `best.pt`, que representa el modelo con el mejor rendimiento en el conjunto de validación.

5.4 Optimización para Despliegue

El modelo `best.pt` se exportó al formato TorchScript (`best.torchscript`) usando la función `model.export()`. Este formato serializa el grafo computacional del modelo, eliminando la dependencia de la librería *ultralytics* y gran parte del código Python, lo que resulta en una reducción significativa del tamaño del modelo y un aumento en la velocidad de inferencia.

5.5 Despliegue y Validación

Se desarrollaron *scripts* de inferencia en Python puro utilizando `torch` y `OpenCV`. Estos *scripts* cargan el modelo `.torchscript`, pre-procesan la imagen de entrada (cámara o archivo) y ejecutan la detección, midiendo el

tiempo de inferencia. La etapa final consiste en integrar esta lógica en una interfaz gráfica con `Tkinter` para su operación en la Jetson Nano.

6 Background de los Modelos Utilizados

6.1 YOLOv10 (You Only Look Once v10)

YOLOv10 es una evolución de la familia de detectores de objetos en una sola pasada. A diferencia de sus predecesores, introduce mejoras clave para la eficiencia, especialmente relevantes para dispositivos de borde.

6.1.1. Descripción Técnica

- **Arquitectura NMS-Free:** Su principal innovación es una arquitectura que no requiere el paso de Non-Maximum Suppression (NMS) durante la post-producción. NMS es un algoritmo que filtra las cajas de detección redundantes y puede ser un cuello de botella computacional. Al eliminarlo, YOLOv10 reduce la latencia de inferencia.
- **Supervisión Dual:** Durante el entrenamiento, emplea una estrategia de "dual label assignments", que proporciona una supervisión más rica y eficiente al modelo, mejorando la precisión sin añadir costo de inferencia.
- **Backbone Eficiente:** Conserva un *backbone* ligero y eficiente basado en la tecnología de Cross-Stage Partial networks (CSP), similar a YOLOv8, que optimiza el flujo de gradientes y reduce la cantidad de parámetros.

6.1.2. Relevancia para el Proyecto

Se eligió **YOLOv10s** por ser una arquitectura de vanguardia diseñada para la eficiencia. Su naturaleza *NMS-free* y su bajo número de parámetros lo convierten en el candidato ideal para alcanzar un rendimiento en tiempo real en la NVIDIA Jetson Nano.

6.2 TorchScript

TorchScript es una herramienta dentro de PyTorch que permite la transición de un modelo desde un entorno de investigación flexible a un entorno de producción de alto rendimiento.

6.2.1. Descripción Técnica

Es un subconjunto de Python que permite crear modelos serializables y optimizables. El compilador de TorchScript analiza el código PyTorch para generar una representación intermedia del grafo computacional. Este grafo puede ser optimizado (ej. fusionando operaciones) y ejecutado de forma independiente en entornos que no tienen Python, como aplicaciones en C++ o en dispositivos móviles.

6.2.2. Relevancia para el Proyecto

Es la tecnología clave que permite el despliegue en la Jetson. Al convertir el modelo `.pt` a `.torchscript`, reducimos la sobrecarga del intérprete de Python, minimizamos las dependencias de software en el dispositivo de borde y aprovechamos las optimizaciones de PyTorch para una inferencia más rápida.

7 Experimental Set-Up

7.1 Entorno de Desarrollo (Nube)

- **Plataforma:** Notebooks de Kaggle.
- **Hardware:** GPU NVIDIA Tesla T4/P100 (para entrenamiento).
- **Software y Librerías:**
  - Python 3.11.13
  - `ultralytics==8.3.169`
  - `torch==2.6.0+cu124`
  - `roboflow`, `opencv-python`, `numpy`, `matplotlib`

7.2 Entorno de Despliegue (Edge)

- **Hardware:** NVIDIA Jetson Nano (4GB), CPU Quad-core ARM A57 @ 1.43GHz, GPU 128-core Maxwell.
- **Software:**
  - NVIDIA JetPack 4.6
  - Python 3.6
  - PyTorch 1.10

- OpenCV 4.5

- **Librerías:** `TorchScript`, `Tkinter`.

7.3 Pipeline Experimental

El flujo de trabajo fue el siguiente:

1. El *dataset* de 933 imágenes (2 clases) se descargó de Roboflow.
2. El modelo `yolov10s.pt` fue entrenado por 100 épocas en Kaggle.
3. El mejor modelo (`best.pt`) fue exportado a `best.torchscript`.
4. La inferencia fue evaluada en la Jetson Nano usando un umbral de confianza (*confidence_threshold*) de 0.5.

8 Resultados y Discusión

8.1 Resultados Cuantitativos

El rendimiento del sistema fue evaluado contra las métricas objetivo, obteniendo resultados satisfactorios en todas las áreas.

Cuadro 1: Resultados Cuantitativos del Sistema

Métrica	**Valor Obtenido**	**Objetivo**	**Estado**
Precisión (validación)	92.1 %	$> 85\,\%$	Superado
Tiempo de Inferencia	380 ms	< 500 ms	Cumplido
Tamaño del Modelo	14 MB	< 20 MB	Cumplido

El modelo no solo cumple sino que supera el umbral de precisión deseado. El tiempo de inferencia en la Jetson Nano se encuentra por debajo del medio segundo, lo que permite una interacción fluida y en tiempo casi real. El tamaño del modelo final (14 MB) es extremadamente bajo, validando la elección de YOLOv10s y la efectividad de la exportación a TorchScript.

8.2 Discusión

8.2.1. Fortalezas

La principal fortaleza del proyecto es haber logrado un alto rendimiento en un **hardware muy restringido**. Esto demuestra la viabilidad de implementar soluciones de IA avanzadas en el borde. El *pipeline* de trabajo es completamente reproducible, desde la descarga de datos hasta la inferencia final.

8.2.2. Limitaciones y Desafíos

- **Sensibilidad a la Iluminación:** La limitación más notable es la sensibilidad del modelo a condiciones de iluminación no vistas en el entrenamiento (ej. contraluz fuerte, sombras faciales pronunciadas o baja luz). Esto puede llevar a una caída en la confianza de la detección o a fallos en la misma.
- **Distorsión de Aspecto:** La función de preprocesamiento (`preprocess_image`) utiliza `cv2.resize` para forzar la imagen a 640x640, lo que distorsiona la relación de aspecto original. Aunque YOLO es robusto a esto, podría afectar la precisión en cámaras con resoluciones no estándar. Un enfoque con *"letterboxing"* (añadir bandas negras) sería más correcto.
- **Dependencia del Entorno CUDA:** Como se observó en el `ValueError` del *notebook*, el entrenamiento es estrictamente dependiente de un entorno con GPUs NVIDIA y CUDA correctamente configurado, lo que resalta la importancia de plataformas en la nube como Kaggle o Colab para esta fase.

Discusión:

- El modelo superó las métricas objetivo en validación
- En Jetson, el consumo de RAM permite ejecución concurrente
- Limitación: Sensibilidad a cambios de iluminación

9 Conclusiones y Trabajo Futuro

9.1 Conclusiones

Se concluye exitosamente que es factible desarrollar e implementar un sistema de control de acceso basado en reconocimiento facial con una **precisión del 92.1 %** en un dispositivo de borde como la Jetson Nano. La metodología de *fine-tuning* sobre YOLOv10 y la optimización del modelo mediante TorchScript fueron estrategias clave para alcanzar los objetivos de rendimiento (**inferencia de 380 ms**) y eficiencia (**modelo de 14 MB**). El proyecto demuestra un ciclo de vida completo de un producto de Machine Learning, desde la concepción y entrenamiento hasta el despliegue optimizado en hardware final.

9.2 Trabajo Futuro

- **Optimización con TensorRT:** Migrar el modelo de TorchScript al motor de inferencia NVIDIA TensorRT. Esto permitiría una optimización más profunda a nivel de *hardware*, aplicando cuantización a FP16 o INT8 y fusionando capas de manera más agresiva para reducir la latencia de inferencia por debajo de los 100 ms.
- **Módulo Anti-Spoofing:** Desarrollar e integrar un segundo modelo clasificador liviano que analice la región del rostro detectada para prevenir ataques de suplantación (ej. mostrar una foto o un video a la cámara). Este módulo podría analizar la textura, el parpadeo o micromovimientos para determinar si el rostro es real.
- **Mejora del Dataset y Robustez:** Ampliar el *dataset* de entrenamiento con imágenes capturadas bajo condiciones de iluminación y ángulos más diversos y extremos. Adicionalmente, implementar un preprocesamiento con *letterboxing* para preservar la relación de aspecto y mejorar la robustez del modelo ante diferentes fuentes de video.

10 Referencias

- *Ultralytics. (2024). YOLOv10 Documentation.*
- *Roboflow. (2023).* Synthetic Data Generation Guide.
- *NVIDIA. (2022). Jetson Nano Developer Kit User Guide.*
- Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems.*
- Wang, A., et al. (2024). YOLOv10: Real-Time End-to-End Object Detection. *arXiv preprint arXiv:2405.14458.*
- CENTURIÓN TALAVERA, Diego Fabián; ALMEIDA DELGADO, Carlos Domingo.*R*econocimiento facial utilizando redes neuronales artificiales en Raspberry Pi.FPUNE Scientific, [S.l.], n. 16, sep. 2022. ISSN 2313-4135.
- Vilchez, Jeferson & Saldarriaga, Darlin. (2024).*A*plicación de algoritmos de redes neuronales y aprendizaje profundo para el reconocimiento facial en el control de asistencia Una revisión de literatura.

- Reaño, R., Carrión, P. and Mansilla, J.-P. (2023).*A*ccess Control Using Facial Recognition with Neural Networks for Restricted Zones.