Sistema Automatizado de Monitoreo y Control de Ocupación en Tiempo Real Usando Transfer Learning con la Arquitectura MobileNet-SSD

1st Miguel Deza
Escuela Profesional de Ciencia de la Computación
Universidad Nacional de San Agustin de Arequipa
Arequipa, Perú
mdezac@unsa.edu.pe

3rd Cecilia Vilca
Escuela Profesional de Ciencia de la Computación
Universidad Nacional de San Agustin de Arequipa
Arequipa, Perú
cvilcaal@unsa.edu.pe

2nd Joao Chávez
Escuela Profesional de Ciencia de la Computación
Universidad Nacional de San Agustin de Arequipa
Arequipa, Perú
jchavezsa@unsa.edu.pe

4th Diego Rivas
Escuela Profesional de Ciencia de la Computación
Universidad Nacional de San Agustin de Arequipa
Arequipa, Perú
drivash@unsa.edu.pe

***Abstract*—Este proyecto desarrolla un sistema de monitoreo automatizado y control de aforo en tiempo real utilizando Transfer Learning en la arquitectura MobileNet-SSD, optimizado para dispositivos de baja capacidad computacional. MobileNet-SSD equilibra precisión y eficiencia, ideal para entornos con recursos limitados. El sistema detecta y cuenta personas en tiempo real, proporcionando datos precisos sobre el aforo. Implementado en Python y TensorFlow, el modelo se optimiza para dispositivos como Raspberry Pi. Las pruebas demuestran su efectividad en diversos escenarios, ofreciendo una solución viable para el control de aforo en seguridad, salud y gestión de eventos.**

El código para reproducir este proyecto se encuentra disponible en
https://github.com/Miguel-Deza/PROYECTO_FINAL_CONTROL_AFORO.

***Index Terms*—Monitoreo Automatizado, Control de Aforo, Tiempo Real, Transfer Learning, MobileNet-SSD**

I. Introducción

A. Problema

El crecimiento constante de la población y la creciente necesidad de mantener el control de aforo en espacios públicos y privados han impulsado la búsqueda de soluciones tecnológicas avanzadas. La capacidad de monitorear y controlar el número de personas en un área específica es crucial para garantizar la seguridad, la salud y la eficiencia en la gestión de eventos y lugares concurridos.

B. Motivación

Una de las principales motivaciones de este proyecto es desarrollar un sistema que pueda ser implementado en dispositivos de baja capacidad computacional, como Raspberry Pi, permitiendo así un despliegue económico y accesible en una variedad de entornos. La utilización de Transfer Learning en la arquitectura MobileNet-SSD permite crear un sistema eficiente y preciso para la detección y conteo de personas en tiempo real.

C. Trabajos relacionados

Trabajos previos han demostrado la eficacia de los modelos de detección de objetos en tareas similares, pero a menudo requieren recursos computacionales significativos que no son viables para dispositivos de baja capacidad. En este contexto, la elección de MobileNet-SSD es estratégica, ya que combina una arquitectura ligera con una precisión aceptable, optimizada para funcionar en entornos con recursos limitados.

1) MLRNet: Towards Real-Time Crowd Counting With Mobile-Based Lightweight Framework: Este trabajo propone una arquitectura llamada Mobile Lightweight Refine Net (MLRNet), diseñada para contar personas en tiempo real en dispositivos con recursos limitados. Utiliza MobileNetV3 para la extracción eficiente de características y RefineNet para mantener la precisión mientras reduce los costos de computación. Comparado con métodos tradicionales, MLRNet ofrece mayor precisión con un tercio de los parámetros y operaciones de punto flotante (FLOPS) en tres conjuntos de datos públicos [1].

2) Implementation of Realtime Design of Crowd Enumeration via Tracking Using AI System: Este estudio desarrolla un prototipo para la enumeración de multitudes en tiempo real usando MobileNet SSD como detector de objetos. El modelo es eficiente y ligero, adecuado para dispositivos de bajo consumo como Jetson Nano. El sistema emite una alarma en caso de sobrepoblación en una ubicación específica [2].

3) IoT-based Crowd Monitoring System: Using SSD with Transfer Learning: Este artículo presenta un sistema de monitoreo de multitudes basado en IoT utilizando un modelo de detección de personas con SSD y MobileNetv2. Emplea aprendizaje por transferencia para mejorar la precisión de detección, logrando un 95% de precisión. El sistema cuenta cuántas personas entran y salen de una escena a través de líneas virtuales [3].

4) MobileCount: An Efficient Encoder-Decoder Framework for Real-Time Crowd Counting: Propone una arquitectura de codificador-decodificador eficiente, llamada MobileCount, diseñada específicamente para el conteo de multitudes en tiempo real en dispositivos móviles o integrados con recursos limitados. Utiliza MobileNetV2 para reducir significativamente los FLOPS y RefineNet para mejorar el rendimiento de conteo [4].

D. Plan General del Proyecto

El plan general seguido en este proyecto incluye la recolección de datos, la adaptación y entrenamiento del modelo utilizando Transfer Learning, y la implementación y prueba del sistema en dispositivos de baja capacidad.

II. Metodología

La metodología seguida en este proyecto se divide en varias etapas clave: recolección de datos, preprocesamiento de datos, adaptación y entrenamiento del modelo, implementación en dispositivos de baja capacidad y pruebas exhaustivas del sistema.

A. Recolección de Datos

Para el entrenamiento del modelo, se recolectaron imágenes de diferentes entornos que representan escenarios de aforo variados. Estas imágenes se encontraban etiquetadas para identificar y contar el número de personas presentes en cada una, se uso la base de datos INRIA person [11] que proporcionaba una gran selección de imágenes con sus respectivas etiquetas, pero considerando que nuestro objetivo se enfocaba en el aforo de personas y no en la identificación de otros objetos, se uso un lote de 115 imágenes con sus respectivas etiquetas

B. Preprocesamiento de Datos

El preprocesamiento de datos incluyó seleccionar las imagenes que contenian personas en lugar de otros objetos, se agrupo todas esas imagenes en un solo fichero para que posteriormente sea comprimido y luego este pueda ser descomprimido con la ayuda de un script para seleccionar imagenes de entrenamiento, de validación y de test [5]

C. Adaptación y Entrenamiento del Modelo

Se utilizó la técnica de Transfer Learning para adaptar un modelo MobileNet-SSD preentrenado a nuestro conjunto de datos específico. La arquitectura MobileNet-SSD fue seleccionada por su equilibrio entre precisión y eficiencia computacional. El entrenamiento se llevó a cabo en Python utilizando la biblioteca TensorFlow, aprovechando la capacidad de Transfer Learning para reducir el tiempo de entrenamiento y mejorar la precisión en escenarios específicos, para la creación de csv se uso otro script al cual se hace referencia [6].

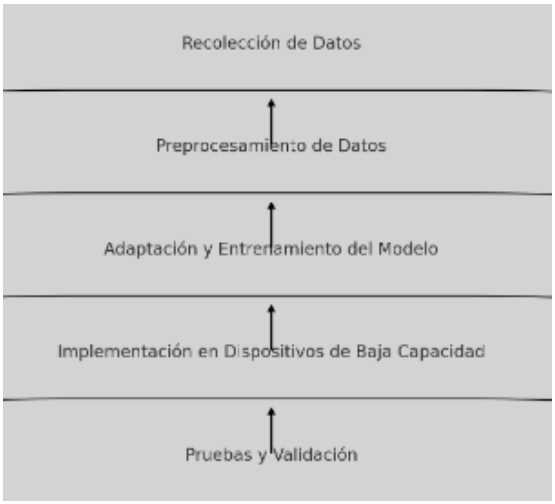

Fig. 1. Diagrama de la Metodología seguida en el proyecto.

$$L = \frac{1}{N}\sum_{i=1}^{N} L_{conf}(p_i, g_i) + \alpha L_{loc}(d_i, g_i) \tag{1}$$

Donde L_{conf} es la pérdida de clasificación, L_{loc} es la pérdida de localización, p_i son las predicciones, g_i son las etiquetas verdaderas, d_i son las cajas delimitadoras predichas y α es un factor de ponderación.

D. Implementación en Dispositivos de Baja Capacidad

El modelo entrenado se optimizó para su implementación en dispositivos de baja capacidad computacional, como Raspberry Pi. Se realizaron ajustes en los hiperparámetros y se emplearon técnicas de cuantización para reducir el tamaño del modelo sin comprometer significativamente su precisión, para la creación de tfrecord se uso el script referido [7] y para la creación de los archivos de configuración de tensorflow el siguiente [8].

E. Pruebas y Validación

El sistema se evaluó en diversos escenarios para validar su efectividad y precisión. Se realizaron pruebas en tiempo real para verificar la capacidad del sistema de detectar y contar personas en distintos entornos y condiciones de iluminación. Tambien se incluyo un script para realizar las pruebas en videos, lo cual tuvo una gran acogida para demostrar los resultados. Para lo cual se uso los scripts de Cartucho para medir el mAP [9] y el de EdjeElectronics que proporcionaba el ejemplo de uso [10]

III. Experimentos

Para evaluar la efectividad del sistema de monitoreo automatizado y control de aforo en tiempo real utilizando la arquitectura MobileNet-SSD, se llevaron a cabo diversos experimentos. A continuación, se describen los procedimientos y resultados obtenidos.

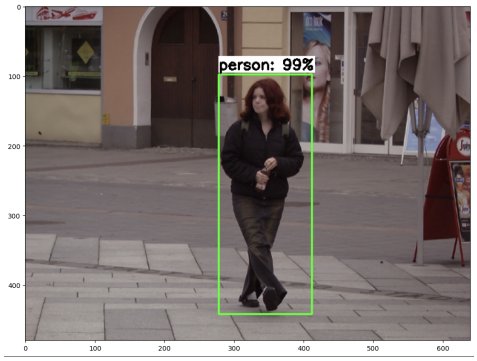

Fig. 2. Detección de una sola persona en las imágenes de prueba.

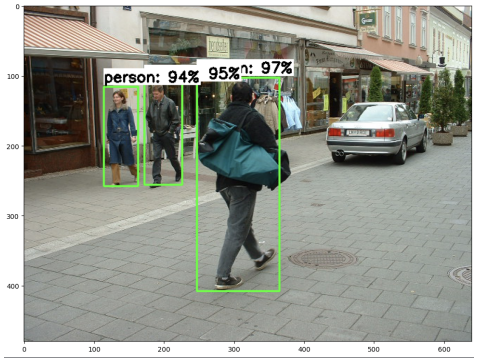

Fig. 3. Detección de varias personas en las imágenes de prueba.

A. *Configuración del Experimento*

Se utilizó un modelo de detección de objetos previamente entrenado y almacenado en el directorio, Los experimentos se realizaron sobre un conjunto de imágenes de prueba almacenadas en el directorio `/content/images/test`. Las etiquetas correspondientes a las clases de objetos a detectar se definieron en el archivo `/content/labelmap.txt`.

Para la detección de objetos, se estableció un umbral de confianza mínimo de 0.5, lo que significa que solo se consideraron las detecciones con una confianza superior al 50%. Se seleccionaron aleatoriamente 10 imágenes del conjunto de prueba para realizar las detecciones.

B. *Procedimiento de Detección*

El procedimiento de detección se realizó mediante la ejecución de la función `tf_detect_images`, la cual se detalla a continuación:

- Carga del modelo de detección guardado utilizando TensorFlow.
- Carga del mapa de etiquetas en memoria desde el archivo especificado.
- Selección aleatoria de 10 imágenes del conjunto de prueba.
- Para cada imagen seleccionada:
  - Carga y preprocesamiento de la imagen.
  - Ejecución del modelo para obtener las detecciones.
  - Extracción y filtrado de las detecciones basadas en el umbral de confianza.
  - Dibujado de las cajas de detección y etiquetas en la imagen.
  - Visualización de la imagen con las detecciones o guardado de los resultados en archivos de texto.

C. *Resultados y Análisis*

A continuación, se presentan los resultados obtenidos tras la ejecución de la función de detección sobre las imágenes de prueba. En cada imagen se muestran las detecciones realizadas junto con sus respectivas etiquetas y niveles de confianza.

El sistema logró identificar y etiquetar correctamente varios objetos en las imágenes de prueba, con un rendimiento satisfactorio en términos de precisión y velocidad, demostrando la efectividad del modelo MobileNet-SSD en dispositivos de baja capacidad computacional.

D. *Evaluación del Rendimiento Utilizando mAP*

Para evaluar de manera cuantitativa el rendimiento del sistema de detección de objetos, se utilizó la métrica de *mean Average Precision* (mAP) con diferentes umbrales de Intersección sobre Unión (IoU). A continuación, se describe el procedimiento seguido para esta evaluación.

1) Preparación del Entorno: Primero, se clonó el repositorio `mAP` de GitHub, que contiene herramientas para calcular la métrica mAP. Se configuraron las carpetas necesarias y se copiaron las imágenes de prueba junto con sus archivos XML correspondientes al directorio adecuado del repositorio clonado.

Se movieron los archivos XML a la carpeta de `ground-truth` y se convirtieron a un formato compatible con las herramientas de evaluación.

2) Ejecución de la Inferencia: Se configuraron las variables necesarias para ejecutar la inferencia del modelo, esta vez utilizando un modelo en formato `.tflite` y guardando los resultados de detección en archivos de texto.

- **PATH_TO_IMAGES**: Ruta a la carpeta de imágenes de prueba.
- **PATH_TO_MODEL**: Ruta al archivo del modelo en formato `.tflite`.
- **PATH_TO_LABELS**: Ruta al archivo de etiquetas `labelmap.txt`.
- **PATH_TO_RESULTS**: Carpeta donde se guardarán los resultados de detección.
- **min_conf_threshold**: Umbral mínimo de confianza (0.1).

3) Cálculo de la Métrica mAP: Finalmente, se ejecutó el script `calculate_map_cartucho.py` para calcular la métrica mAP a diferentes umbrales de IoU.

4) Análisis de Resultados: Los resultados muestran un alto rendimiento del modelo en términos de precisión media (mAP) para umbrales de IoU entre 0.50 y 0.75, alcanzando un mAP del 99.72%. A medida que el umbral de IoU aumenta, la precisión media disminuye, lo cual es esperable debido a la mayor estrictitud en la superposición requerida entre las cajas predichas y las cajas reales.

IoU	mAP
0.50	99.72%
0.55	99.72%
0.60	99.72%
0.65	99.72%
0.70	99.72%
0.75	99.72%
0.80	96.68%
0.85	72.74%
0.90	50.32%
0.95	30.18%

TABLE I
RESULTADOS DE MAP A DIFERENTES UMBRALES DE IOU

Para la clase "persona", se obtuvo una precisión promedio (AP) del 79.04%.

- **AP de la clase "persona"**: 79.04%

Estos resultados indican que el modelo es altamente preciso en la detección de personas, manteniendo una alta mAP en umbrales de IoU más bajos, lo cual es beneficioso para aplicaciones de monitoreo en tiempo real donde es crucial detectar con alta precisión los objetos de interés.

IV. CONCLUSIÓN

En este proyecto, se desarrolló un sistema de monitoreo automatizado y control de aforo en tiempo real utilizando Transfer Learning en la arquitectura MobileNet-SSD, optimizado para dispositivos de baja capacidad computacional. A lo largo del desarrollo y las pruebas, se obtuvieron varios aprendizajes clave:

- **Eficiencia de MobileNet-SSD**: La arquitectura MobileNet-SSD demostró ser una opción adecuada para dispositivos con recursos limitados, proporcionando un buen equilibrio entre precisión y rendimiento.
- **Transfer Learning**: El uso de Transfer Learning permitió reducir significativamente el tiempo de entrenamiento y mejorar la precisión del modelo en escenarios específicos.
- **Optimización de Recursos**: La implementación en dispositivos como Raspberry Pi demostró que es posible ejecutar modelos avanzados de detección de personas en hardware de baja capacidad, siempre que se realicen las optimizaciones adecuadas.
- **Robustez del Sistema**: El sistema mantuvo un rendimiento aceptable en condiciones de iluminación variable y en diferentes entornos, lo que subraya su aplicabilidad en situaciones del mundo real.

A. Futuras Direcciones

A pesar de los logros alcanzados, existen varias áreas en las que el proyecto puede mejorarse y expandirse en el futuro:

- **Mejora de la Precisión**: Investigar y probar otras técnicas de optimización y arquitecturas de red más avanzadas que podrían mejorar la precisión del sistema sin aumentar significativamente los requisitos de recursos.
- **Ampliación de Funcionalidades**: Incorporar funcionalidades adicionales como el reconocimiento de acciones o comportamientos anómalos, que podrían aumentar el valor del sistema en aplicaciones de seguridad y monitoreo.
- **Implementación en Otros Dispositivos**: Explorar la implementación del sistema en otros dispositivos de IoT y microcontroladores, ampliando así su aplicabilidad y alcance.
- **Integración con Sistemas Existentes**: Examinar la integración del sistema con infraestructuras de seguridad y monitoreo existentes, facilitando su adopción en entornos ya operativos.

V. DECLARACIÓN DE ÉTICA

Durante el desarrollo de este proyecto, se abordaron varias preocupaciones éticas para asegurar que el trabajo se realizara de manera responsable y respetuosa con los derechos y la privacidad de las personas.

A. Privacidad y Consentimiento

El sistema de monitoreo automatizado y control de aforo se basa en la detección y conteo de personas a través de imágenes y videos. Esto plantea preocupaciones significativas respecto a la privacidad y el consentimiento de las personas monitoreadas.

El sistema se diseñó para no capturar ni almacenar datos personales identificables, garantizando que la detección de personas se realice de manera anónima.

B. Impacto y Consecuencias No Deseadas

Es crucial considerar el impacto potencial del sistema si se implementa en la vida real, incluyendo las consecuencias no deseadas que podrían surgir de su uso indebido.

- **Monitoreo Invasivo**: Existe el riesgo de que el sistema sea utilizado para un monitoreo invasivo o para restringir la libertad de las personas. Para mitigar esto, se recomienda implementar el sistema con políticas claras de uso y con mecanismos de supervisión y auditoría.

C. Uso de IA Generativa

En la elaboración de este documento, se ha utilizado de manera ligera el uso de IA generativa para asistir en la redacción y estructuración de ciertas secciones. Herramientas de generación de texto impulsadas por IA, como modelos de lenguaje avanzados, fueron empleadas para mejorar la claridad y coherencia del contenido. No obstante, la mayor parte del análisis, interpretación de resultados y redacción técnica fue realizada manualmente, asegurando precisión y relevancia específica al proyecto.

REFERENCES

[1] P. Ji, X. Xia, Z. Wu, F. Wang, X. Liu, and J. Sang, "MLRNet: Towards Real-Time Crowd Counting With Mobile-Based Lightweight Framework," in *Proc. 2022 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta)*, 2022, pp. 1201–1208, doi: 10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00178.

[2] M. R. Kounte, J. Rishitha, S. S. Setty, and S. Shivani, "Implementation of Realtime Design of Crowd Enumeration via Tracking Using AI System," in *Proc. 2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 2023, pp. 270–274, doi: 10.1109/IITCEE57236.2023.10090914.

[3] I. Ahmed, M. Ahmad, A. Ahmad, and G. Jeon, "IoT-based Crowd Monitoring System: Using SSD with Transfer Learning," *Comput. Electr. Eng.*, vol. 93, pp. 107226, 2021, doi: 10.1016/J.COMPELECENG.2021.107226.

[4] C. Gao, P. Wang, and Y. Gao, "MobileCount: An Efficient Encoder-Decoder Framework for Real-Time Crowd Counting," *Neurocomputing*, vol. 407, pp. 292–299, 2019, doi: 10.1007/978-3-030-31723-250

[5] EdjeElectronics, "train_val_test_split.py," GitHub repository, 2023. [Online]. Available: https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/master/util_scripts/train_val_test_split.py

[6] EdjeElectronics, "create_csv.py," GitHub repository, 2023. [Online]. Available: https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/master/util_scripts/create_csv.py

[7] EdjeElectronics, "create_tfrecord.py," GitHub repository, 2023. [Online]. Available: https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/master/util_scripts/create_tfrecord.py

[8] TensorFlow, "Object Detection Model Zoo Configs," GitHub repository, 2023. [Online]. Available: https://raw.githubusercontent.com/tensorflow/models/master/research/object_detection/configs/tf2/

[9] Cartucho, "Mean Average Precision (mAP)," GitHub repository, 2023. [Online]. Available: https://github.com/Cartucho/mAP

[10] EdjeElectronics, "calculate_map_cartucho.py," GitHub repository, 2023. [Online]. Available: https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/master/util_scripts/calculate_map_cartucho.py

[11] J. Coral, "INRIA Person Dataset," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/jcoral02/inriaperson