

Sistema de Clasificación Multiclase de Eventos de Riesgo en Ciclismo Urbano mediante Redes Neuronales

Carlos David Ramírez Muñoz

Resumen — El ciclismo urbano es un pilar fundamental de la movilidad sostenible; sin embargo, los biciusuarios enfrentan una alta siniestralidad vial, evidenciada por 69 fallecidos y más de 2.000 lesionados en Bogotá durante 2024. Los sistemas de monitoreo actuales, limitados por bajas frecuencias de geolocalización (~1 Hz), resultan insuficientes para categorizar la complejidad de los eventos de seguridad. Este trabajo propone una arquitectura integral de Internet de las Cosas (IoT) de cuatro capas (percepción, comunicación, backend y visualización) fundamentada en el paradigma de Edge Intelligence. El sistema emplea nodos IoT basados en microcontroladores ESP32 y dispositivos inteligentes para capturar telemetría inercial de alta frecuencia (50 Hz) y datos GPS. Mediante técnicas de TinyML, se implementan redes neuronales para la clasificación multiclase de eventos críticos de seguridad (CSE, por sus siglas en inglés), tales como frenados bruscos e impactos por baches, permitiendo su ejecución local en el borde. La solución busca integrar un flujo sincronizado de datos hacia la plataforma ThingsBoard a través de protocolos ligeros, facilitando la generación de inteligencia geoespacial y mapas de calor para la identificación precisa de riesgos viales y la planificación urbana resiliente.

Abstract — Urban cycling is a fundamental pillar of sustainable mobility; however, cyclists face high road casualty rates, evidenced by 69 fatalities and over 2,000 injuries in Bogotá during 2024. Current monitoring systems, limited by low geolocation frequencies (~1 Hz), are insufficient to categorize the complexity of safety events. This work proposes a comprehensive four-layer Internet of Things (IoT) architecture (perception, communication, backend, and visualization) based on the Edge Intelligence paradigm. The system employs IoT nodes based on ESP32 microcontrollers and smart devices to capture high-frequency inertial telemetry (50 Hz) and GPS data. Through TinyML techniques, neural networks are implemented for the multiclass classification of Critical Safety Events (CSE), such as harsh braking and pothole impacts, enabling local execution at the edge. The solution integrates a synchronized data flow to the ThingsBoard platform via lightweight protocols, facilitating the generation of geospatial intelligence and heatmaps for precise road risk identification and resilient urban planning.

I. Introducción

La bicicleta representa actualmente un medio de transporte alternativo, ecológico, descongestionante de tráfico y mitigador de enfermedades cardiovasculares. Sin embargo, los ciclistas urbanos han tenido que enfrentar condiciones adversas de seguridad vial con preocupantes índices de víctimas de siniestros viales. Para la ciudad de Bogotá, se presentaron 69 fallecidos y 2049 lesionados en 2024, lo que los hace el tercer tipo de actor vial más vulnerable (después de motociclistas y peatones) [1], sumado a la percepción ciudadana

general de inseguridad e incidentes leves no contempladas en estas cifras.

Ante este panorama es fundamental la creación de mecanismos para la identificación y mitigación de eventos de riesgo de los ciclistas. Los sistemas de monitoreo actuales suelen usar GPS de baja frecuencia (~1 Hz), mediante el uso de smartphones por ejemplo, que solo permiten análisis de trayectorias, o modelos binarios de detección de anomalías (evento normal vs. anomalía). Sin embargo, para mejorar la seguridad vial, puede ser sumamente útil ir más allá de la detección binaria y clasificar la naturaleza específica del evento (ej. frenado de emergencia, esquivar lateral por obstáculo, o impacto por estado del pavimento, entre otros).

El problema radica en la alta complejidad y no linealidad de las señales inerciales generadas durante la conducción. Este proyecto aborda dicho problema mediante sensores inerciales integrados en smartphone, específicamente acelerómetro y giroscopio, capturados mediante la aplicación Sensor Logger y re muestreados a 50 Hz para homogeneizar el análisis. A partir de estas señales se implementaron modelos neuronales shallow y deep para clasificar tipos de incidentes, mientras que el GPS se utilizó como fuente de georreferenciación para crear mapas de calor de distintas zonas de la ciudad y los incidentes que ocurrieron en estas durante la recopilación del dataset propio.

II. OBJETIVOS

Objetivo General:

Diseñar, crear y evaluar un sistema de clasificación multiclase para categorizar eventos críticos, anomalías en el terreno y maniobras de riesgo en el ciclismo urbano utilizando datos de geolocalización y de telemetría inercial, y luego utilizar este dataset para generar mapas de riesgo que ilustren de manera simple el conocimiento obtenido sobre la ciudad.

Objetivos Específicos:

- **Recolección y preprocesamiento de datos:** Consolidar un conjunto de datos propio con la herramienta Sensor Logger, sincronizando señales inerciales, remuestreando a 50 Hz, corrigiendo anotaciones manuales y segmentando las señales en ventanas temporales.
- **Reconocimiento básico de eventos (Creación Red Neuronal Poco Profunda):** Desarrollar una arquitectura de red neuronal superficial basada en un perceptrón multicapa utilizando técnicas previas de Feature Engineering como extraer valores estadísticos (media, varianza, picos) de las ventanas de tiempo inerciales para clasificar de manera básica los eventos abruptos.
- **Reconocimiento avanzado de eventos (Creación Red Neuronal Profunda):** Implementar y evaluar una red neuronal profunda basada en capas convolucionales 1D (CNN 1D), entrenada directamente sobre ventanas crudas de acelerómetro y giroscopio. Las coordenadas GPS no se usan como entrada de la CNN, sino como información posterior para georreferenciar predicciones y construir mapas de riesgo.
- **Evaluación Comparativa:** Analizar el desempeño de ambas arquitecturas neuronales utilizando métricas de clasificación multiclase (Accuracy, Precision, Recall y F1-Score)

empleando como instrumento clave las matrices de confusión para evaluar la capacidad de los modelos para distinguir los diferentes tipos de anomalías.

- **Identificación de zonas de riesgo:** Integrar los resultados obtenidos de la clasificación de los eventos peligrosos junto a las coordenadas GPS para generar mapas de calor que evidencien los tramos viales con mayor recurrencia de accidentes.

III. REVISIÓN DE LITERATURA

A. Seguridad Vial y Maniobras de Detección Basadas en Sensores Inerciales

El uso de sensores inerciales como el acelerómetro y el giroscopio, disponibles en smartphones, se ha convertido en una solución de bajo costo y escalable para análisis de seguridad vial. Estos sensores permiten la detección de patrones en la conducción, como el frenado, los cambios de carril, y maniobras agresivas sin necesidad de hardware vehicular especializado. En concreto la aplicación Sensor Logger permite integrar este tipo de sensores junto a la ubicación GPS de dispositivos Android e Iphone, pudiendo generar conjuntos de datos sincronizados y con preprocesamiento y calibración básicas.

B. Recolección de Datos y Preprocesamiento

El procesamiento de señales inerciales requiere una etapa de pretratamiento rigurosa para garantizar consistencia y estabilidad en los modelos de clasificación. En sistemas basados en smartphones o IMUs, es común aplicar:

- Sincronización temporal de señales multicanal
- Filtrado para reducción de ruido
- Eliminación de componente gravitacional
- Normalización o estandarización
- Segmentación en ventanas temporales deslizantes

La segmentación por ventanas es una estrategia ampliamente utilizada en reconocimiento de actividades humanas, ya que permite transformar señales continuas en muestras etiquetadas para aprendizaje supervisado [4]. La elección del tamaño de ventana y el porcentaje de solapamiento impacta directamente el desempeño del modelo.

C. Ingeniería de Características y Modelos Superficiales

Los enfoques tradicionales para clasificación de series temporales inerciales se basan en la extracción manual de características estadísticas a partir de cada ventana temporal. Entre las características más comunes se encuentran:

- Media
- Varianza
- Desviación estándar
- Energía (RMS)
- Amplitud pico-a-pico

Estas características reducen la dimensionalidad del problema antes de aplicar clasificadores como SVM, k-NN o Random Forest. Estudios comparativos en HAR han mostrado que estos métodos pueden alcanzar desempeños competitivos cuando las características están cuidadosamente diseñadas [5].

Las redes neuronales superficiales (MLP con una o dos capas ocultas) representan una extensión de estos enfoques, permitiendo modelar relaciones no lineales entre características.

D. Aprendizaje

El aprendizaje profundo ha demostrado ventajas significativas frente a métodos tradicionales al permitir la extracción automática de representaciones jerárquicas directamente desde datos crudos.

1. Convolutional Neural Networks (CNN)

Las CNN aplicadas a señales temporales (1D-CNN) realizan convoluciones sobre el eje temporal para capturar patrones locales característicos. En HAR, CNN profundas han superado sistemáticamente métodos tradicionales basados en características manuales [4].

En el estudio de da Silva [7], una CNN aplicada al Bike&Safe Dataset [3] logró detectar anomalías en comportamiento ciclista utilizando directamente las secuencias de sensores, demostrando la efectividad del enfoque profundo para análisis de seguridad vial.

2. Long Short-Term Memory (LSTM)

Las redes LSTM, una variante de redes neuronales recurrentes, están diseñadas para capturar dependencias temporales de largo plazo. Estudios comparativos han demostrado que arquitecturas híbridas CNN-LSTM combinan extracción automática de características y modelado secuencial, obteniendo altos niveles de desempeño en clasificación de actividades humanas con datos IMU [6].

E. Evaluación de Modelos de Clasificación Multiclase

En problemas de clasificación multiclase, especialmente cuando existen clases desbalanceadas o eventos poco frecuentes, la evaluación no debe limitarse a la exactitud global. Se suele recomendar el uso de:

- Accuracy
- Precision
- Recall
- F1-Score
- Matriz de confusión

Estas métricas permiten evaluar la capacidad del modelo para distinguir entre clases similares y analizar errores sistemáticos. Estudios en HAR y clasificación de señales inerciales enfatizan la importancia de analizar métricas por clase y promedios macro para garantizar robustez del modelo [4], [5].

F. Eventos de Riesgo y Anomalías

Los eventos de tráfico que requieren maniobras evasivas rápidas (frenado, giro, o en algunos casos ambos) se denominan Critical Safety Events (CSE). Para la realización del proyecto hemos decidido clasificar el los CSE en 4 categorías principales

1. **Eventos Longitudinales (Frenado Brusco):** Estos representan frenados o desaceleraciones bruscas que se evidencian en picos negativos repentinos en el eje Y o X de movimiento (según el montaje). Son detenciones de emergencia ante obstáculos.
2. **Eventos Laterales (Viraje Repentino):** Maniobras rápidas en el eje horizontal para evitar obstáculos, baches, entre otros elementos. Se ve reflejado en el giroscopio y la aceleración lateral
3. **Eventos Verticales (Impactos por Infraestructura):** Anomalías causadas principalmente por el estado deficiente

de la vía (baches, resaltos, grietas). Estos se ven reflejados en picos en el eje Z del acelerómetro.

4. Para efectos de este proyecto, se trabajó con tres tipos de eventos críticos anotados: bache, esquivada de emergencia y frenada de emergencia. La clase normal se incluye como categoría de referencia para el entrenamiento supervisado. La pérdida de control o caída se reconoce como un evento crítico relevante, pero no fue incluida como clase entrenada en esta versión del modelo.

G. Tecnologías Habilitadoras: Hardware de Percepción y Sistemas Embebidos

El nodo sensor del sistema se basa en dos plataformas complementarias. El smartphone (Xiaomi POCO X6) ofrece una IMU MEMS integrada con acelerómetro y giroscopio de tres ejes, GPS y conectividad inmediata, constituyendo el dispositivo de captura principal en la fase actual. El microcontrolador ESP32 representa la alternativa embebida de bajo costo: opera a 240 MHz con 520 KB de SRAM y soporte dual-core, lo que permite ejecutar modelos TinyML de complejidad moderada. Sin embargo, su memoria SRAM limita la ejecución de arquitecturas 1D-CNN completas, favoreciendo el despliegue del MLP cuantizado como alternativa viable para inferencia en el borde. Una Raspberry Pi Zero W, con 512 MB de RAM, permitiría ejecutar modelos más complejos pero a mayor costo energético y económico. Los sensores MEMS (Sistemas Microelectromecánicos) se prefieren en este dominio por su miniaturización, bajo consumo (~1–5 mA) y adecuada resolución para capturar eventos inerciales en el rango de 0–16 g [8].

IV. ESTUDIO DE CASO

La literatura relevante para este proyecto puede organizarse en tres líneas de investigación convergentes: detección de eventos mediante señales inerciales, seguridad ciclista asistida por IoT, y arquitecturas IoT para movilidad urbana inteligente.

A. Reconocimiento de Actividad y Detección de Eventos con IMU

Los trabajos fundacionales en HAR (Human Activity Recognition) establecieron que características estadísticas extraídas sobre ventanas temporales de señales acelerométricas permiten clasificar actividades físicas con alta precisión usando clasificadores tradicionales [5]. Ordóñez y Roggen [4] demostraron posteriormente que arquitecturas profundas CNN-LSTM superan consistentemente estos enfoques sobre señales IMU multicanal de dispositivos vestibles, justificando la progresión hacia 1D-CNN prevista en la segunda fase del presente proyecto. Jaén-Vargas et al. [6] corroboran esta tendencia aplicada específicamente a señales inerciales de smartphones, confirmando la viabilidad del hardware de consumo como plataforma de captura.

B. Seguridad Ciclista: Sistemas Previos y sus Limitaciones: Se han identificado dos investigaciones que representan soluciones paralelas y parciales hacia los objetivos propuestos, constituyendo los antecedentes directos más relevantes. El estudio de Yaqoob et al. [2], "*Detection of anomalies in cycling behavior with convolutional neural network and deep learning*", demuestra la viabilidad de las redes convolucionales para identificar maniobras evasivas ciclistas y CSE en la ciudad de Catania, Italia. El modelo BeST-DAD (Bicyclist Safety Tracking - Deep Anomaly Detection), basado en un autoencoder convolucional, logra una clasificación binaria con F-Score del 77 % y Recall del 100 %. Este estudio justifica las CNN como arquitecturas adecuadas para procesar señales de movilidad; sin embargo, depende exclusivamente de datos de geolocalización de alta

frecuencia —velocidad y dirección— excluyendo los sensores inerciales, lo que limita la clasificación multiclase e identificación granular del tipo de anomalía. Da Silva [3], en "*Enhancing Cycling Safety in Smart Cities: A Data-Driven Embedded Risk Alert System*", implementa un sistema de alerta de riesgo ciclista en tiempo real con hardware embebido de bajo costo (Raspberry Pi Zero), empleando una infraestructura IoT para recolección masiva de datos inerciales. Aunque el sistema demuestra solidez en la captura y transmisión de datos, no incorpora modelos de aprendizaje automático para clasificación, limitándose a detección por umbrales. Este trabajo además provee el dataset Bike&Safe, considerado inicialmente como insumo de entrenamiento para el presente proyecto, aunque descartado posteriormente por incompatibilidad de escala con los datos propios (detallado en Sección 5). En síntesis, el objetivo del proyecto es superar la exclusión de variables cinemáticas de [2] incorporando acelerómetro y giroscopio, mientras se superan las carencias analíticas de [3] mediante redes neuronales supervisadas. Ningún trabajo previo identificado combina simultáneamente captura inercial de alta frecuencia, clasificación multiclase de CSE y arquitectura IoT de cuatro capas con inferencia en el borde.

C. Arquitecturas IoT para Movilidad Urbana

Alavi et al. [12] presentan una revisión comprehensiva de ciudades inteligentes habilitadas por IoT, identificando la combinación de geolocalización con sensores de contexto como paradigma dominante en aplicaciones de seguridad vial. Kaur [14] propone el modelo de referencia de cuatro capas (percepción, comunicación, procesamiento y aplicación) como estructura base para sistemas IoT interoperables, modelo que adopta directamente la arquitectura propuesta en este proyecto. Reka y Dragicevic [13] destacan la necesidad de arquitecturas de borde para reducir latencia en sistemas de alerta en tiempo real, argumento central del paradigma Edge Intelligence adoptado aquí.

V. DISEÑO DE LA ARQUITECTURA IoT PROPUESTA

Arquitectura de Referencia: El diseño sigue el modelo de referencia IoT de cuatro capas propuesto por Kaur [14] y ampliamente adoptado en la literatura: (1) capa de percepción (sensing), (2) capa de red y comunicación, (3) capa de procesamiento y backend, y (4) capa de aplicación y visualización. Este modelo de referencia orienta la interoperabilidad entre componentes heterogéneos y es la base sobre la cual se instancia la arquitectura solución descrita a continuación.

Arquitectura de Solución:

Capa	Descripción
Capa 1 - El Nodo	Captura de datos mediante el dispositivo físico (ESP32 o smartphone usando Sensor Logger). Este analiza los datos localmente y solo emite una alerta cuando detecta un evento crítico. Se contempla la posibilidad de que la tarjeta ESP32 almacene

	de forma local en la tarjeta SD para un posterior envío de los datos cuando se disponga de conexión a internet. También puede ser beneficioso el implementar edge computing, utilizando TensorFlow Lite en la ESP 32 cargando la red pre-entrenada para detectar anomalías ciclistas.
Capa 2 - Comunicación	El nodo almacena los datos localmente (tarjeta SD o memoria flash) y, al detectar conectividad, los transmite al backend mediante MQTT (puerto 1883/8883) con QoS nivel 1 para alertas de eventos críticos, o HTTP POST para lotes de datos históricos. El broker MQTT de Thingsboard actúa como punto de recepción. En condiciones de red intermitente, el nodo opera en modo store-and-forward, priorizando el envío de alertas CSE sobre la telemetría cruda.
Capa 3 - Backend y almacenamiento	Los datos son recibidos por el broker MQTT/API Rest de <i>Thingsboard</i> para que la propia plataforma pueda procesarlos posteriormente. De esta manera, es posible ir enviando los datos de forma progresiva a Thingsboard, bien sean datos crudos o preprocesados con edge computing del ESP32.
Capa 4 - Visualización	Muestra gráficamente los datos recolectados, haciendo uso de mapas de calor, mapas reales mediante GPS y distintos tipos de gráficas que permitan identificar las zonas de la ciudad con el mayor número de anomalías/accidentes

Tabla 1. Arquitectura de Solución

El pipeline de transformación de datos sigue tres niveles:

1. Dato crudo — muestras IMU a 50 Hz y coordenadas GPS a 1 Hz almacenadas en CSV
2. Dato procesado — ventanas de 128 muestras con 72 features estadísticas extraídas y etiqueta CSE asignada por el modelo
3. Información IoT accionable — eventos clasificados georeferenciados, publicados vía MQTT a Thingsboard, donde se agregan en celdas de ~20 m para generar el heatmap de severidad vial. Este heatmap constituye el producto de información de mayor nivel: transforma telemetría inercial individual en inteligencia geoespacial colectiva útil para planificadores urbanos y autoridades de movilidad.

Componentes tecnológicos:

Componente	Tecnología Escogida	Justificación
Nodo sensor (fase actual)	Smartphone Xiaomi POCO X6	IMU MEMS integrada, GPS, conectividad inmediata
Nodo sensor (fase futura)	ESP32 + MPU-6050	Bajo costo, soporte TinyML, despliegue autónomo
Captura de datos (fase actual)	Sensor Logger (Android)	Sincronización multicanal, exportación CSV, anotación en tiempo real
Protocolo de comunicación	MQTT (Thingsboard broker)	Bajo overhead, QoS configurable, soporte nativo en Thingsboard
Backend / Plataforma IoT	Thingsboard Community Edition	API REST + MQTT, dashboards nativos, soporte geoespacial
Modelo de clasificación (borde)	MLP cuantizado / TFLite Micro	Compatible con ESP32, latencia sub-segundo
Modelo de clasificación (nube)	MLP supervisado (Python/Keras)	97.8% accuracy, F1 macro 0.888
Visualización	Folium + Thingsboard Maps	Mapas interactivos, heatmaps, marcadores CSE
Formato de datos	JSON sobre MQTT / CSV local	Interoperabilidad, soporte nativo Thingsboard
Sistema de coordenadas	WGS84 (EPSG:4326)	Estándar geoespacial universal

Tabla 2. Componentes Tecnológicos

VI. PROTOCOLOS DE COMUNICACIÓN Y ESTÁNDARES DE INTEROPERABILIDAD

La intermitencia de conectividad inherente a trayectos ciclistas urbanos —zonas con cobertura degradada o cambios entre Wi-Fi y datos móviles— impone requisitos específicos sobre el protocolo de comunicación. MQTT (Message Queuing Telemetry Transport), protocolo basado en publicación/suscripción sobre TCP con

sobrecarga mínima de cabecera (2 bytes fijos), resulta preferible a HTTP POST para el envío de telemetría desde el nodo hacia Thingsboard. Su soporte nativo de niveles QoS (0, 1 y 2) permite garantizar la entrega de alertas críticas incluso bajo condiciones de red inestables, mientras que el mecanismo de last will notifica al backend ante desconexiones inesperadas del dispositivo [9]. HTTP POST se reserva para transferencias de lotes de datos crudos cuando el nodo dispone de conectividad estable.

En cuanto a estándares de interoperabilidad, la telemetría se serializa en formato JSON, ampliamente soportado por la API REST y el broker MQTT de Thingsboard. Para la capa de visualización geoespacial, los mapas de calor y marcadores de eventos se generan sobre teselas compatibles con el estándar Web Map Tile Service (WMTS), con coordenadas en el sistema de referencia WGS84 (EPSG:4326), garantizando interoperabilidad con plataformas de planificación urbana. El modelo TinyML destinado al ESP32 seguirá el formato TensorFlow Lite Micro (.tflite), estándar de facto para inferencia en microcontroladores de recursos limitados [10].

VII. DATASET DE ENTRENAMIENTO Y NODO INICIAL

Inicialmente se consideró utilizar el dataset Bike&Safe [3], presentado en el artículo "Detection of anomalies in cycling behavior with convolutional neural network and deep learning" [7], en conjunción con datos propios recolectados con Sensor Logger, con el fin de tener una base de entrenamiento sólida que a la vez permitiera hacer validaciones locales diversas. Sin embargo, el análisis exploratorio reveló una incompatibilidad de escala entre ambos conjuntos de datos —explicada en detalle en la sección siguiente—, lo que llevó a la decisión de utilizar únicamente el dataset propio para el entrenamiento, validación y prueba de la red neuronal.

Para la creación del dataset propio se utilizó el smartphone Xiaomi POCO X6, equipado con una unidad de medición inercial (IMU) que incluye acelerómetro MEMS de tres ejes y giroscopio MEMS de tres ejes. La captura se configuró desde la API de sensores de Android y posteriormente las señales fueron remuestreadas a 50 Hz para garantizar una tasa homogénea de análisis. También se usó el GPS del smartphone, con una tasa aproximada de 1 Hz, exclusivamente para asociar coordenadas a ventanas y predicciones.

El dispositivo fue montado en una bolsa de cuadro con una inclinación aproximada de $\sim 40^\circ$ respecto a la vertical, por lo que los ejes individuales del acelerómetro no se alinean con los ejes físicos de la bicicleta. Sin embargo, dado que las features utilizadas se basan en estadísticas de magnitud vectorial —invariantes a rotaciones— y que la posición de montaje se mantuvo constante entre sesiones, esta inclinación no representa una fuente de error sistemático para el modelo.



Fig. 1. Configuración del montaje experimental. El smartphone se fijó al cuadro de la bicicleta con una pequeña maleta sujetada con broches en velcro.

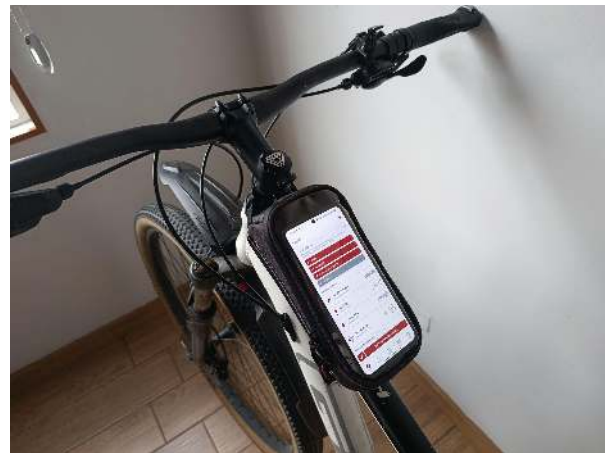


Fig. 2. Detalle del montaje. La configuración permite la utilización de la aplicación sensor logger y la anotación manual de eventos mediante interacciones táctil con el smartphone

Los datos fueron capturados utilizando la aplicación **Sensor Logger**, que permite registrar de forma sincronizada múltiples sensores del dispositivo —acelerómetro lineal, giroscopio, GPS y acelerómetro total— exportando toda la sesión en archivos CSV con marca de tiempo de alta resolución por muestra. Se utilizó el canal de aceleración lineal (accelerometer_x/y/z), el cual corresponde a la aceleración neta descontada la componente gravitacional, con lo que el valor de referencia en reposo es aproximadamente 0 m/s^2 por eje.



Fig. 3. Screenshot de aplicación Sensor Logger. La interfaz de la aplicación sensor logger permite realizar anotaciones manuales a eventos mientras se obtienen los datos inerciales

La recolección de datos siguió una estrategia de dos tipos de recorrido complementarios:

Recorridos artificiales (etiquetados). Se realizaron trayectos de corta duración en entornos controlados, con el propósito de generar eventos de ciclismo de seguridad (CSE) deliberados y correctamente etiquetados. Durante estos recorridos el ciclista ejecutó de forma intencional y repetida tres tipos de evento: impacto con bache (bache), esquivada de emergencia (esquivada) y frenada de emergencia (freno de emergencia). Cada evento fue registrado manualmente en tiempo real mediante la función de anotación de Sensor Logger, que inserta una etiqueta con marca de tiempo en el mismo archivo CSV. Esta modalidad prioriza la densidad de eventos etiquetados por unidad de tiempo sobre la variedad de rutas. Las marcas manuales introducen dos fuentes de error: retardo humano (el botón se pulsa después del evento físico) y toques consecutivos accidentales (varias anotaciones en pocos segundos para el mismo intento). Estos errores fueron mitigados satisfactoriamente mediante estrategias explicadas en la sección 6: Preprocesamiento de datos y análisis exploratorio de datos.

Para este tipo de recorridos se usaron ambientes controlados, trayectorias con forma de circuito y baja afluencia de personas y vehículos. Esto permitió la ejecución deliberada de eventos críticos como esquivadas de emergencia y frenos de emergencia sin presentar un riesgo adicional.



Fig. 4. Mapa de recorrido artificial. Se muestra el recorrido artificial más común, el cual tiene forma de circuito permitiendo capturar datos de forma consistente y controlada



Fig. 5. Fotografía de sección de recorrido artificial. Los recorridos artificiales cuentan con baja afluencia de vehículos y personas lo que permite la realización de maniobras evasivas y frenados de emergencia sin riesgo inminente en el entorno

Recorridos naturales (sin etiquetar). Se realizaron recorridos urbanos de mayor duración en condiciones reales de tráfico, sin etiquetar eventos de forma sistemática. El propósito de estos recorridos es doble: por un lado, aportar ejemplos de comportamiento normal en escenarios de alta varianza (distintas superficies, velocidades y condiciones de tráfico); por otro lado, servir como conjunto de datos para la generación de mapas de calidad de pavimento, registrando de forma natural y no forzada las anomalías presentadas en los recorridos. Al no contar con etiquetas densas, estos datos se utilizan en el modelo heurístico de severidad y como contexto geoespacial, no para el entrenamiento supervisado directo.

Para estos recorridos se tomaron en cuenta varias directrices como circular por la ciclovía o carril bici siempre que esté disponible, transitar a una velocidad moderada y respetando las normas de tránsito, con el fin de obtener de forma más precisa el estado y dinámicas de las vías.

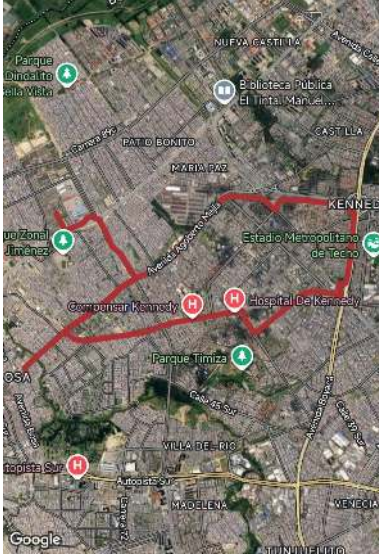


Fig. 6. Mapa de recorrido natural. Los recorridos naturales buscan capturar las características de las ciclovías más usadas de forma natural y espontánea



Fig. 7. Fotografía de sección de recorrido natural. Los recorridos naturales cuentan con diversa afluencia de vehículos y condiciones cambiantes, capturando las características normales de los trayectos

En total, el dataset propio comprende 2.760.216 muestras, 11 recorridos artificiales etiquetados y 37 recorridos naturales, para un total de 40 recorridos y más de 15 horas de grabación. Los recorridos artificiales concentran 335 eventos anotados manualmente: 152 baches, 96 esquivadas y 87 frenadas de emergencia. Los recorridos naturales se utilizaron principalmente para entrenamiento heurístico, evaluación por particiones y generación de mapas de severidad y riesgo.

VIII. PREPROCESAMIENTO DE DATOS Y ANÁLISIS

EXPLORATORIO DE DATOS

1. Incompatibilidad con Dataset Bike&Safe

El análisis exploratorio de magnitudes de aceleración reveló que Bike&Safe reporta valores con mediana $\approx 4,55 \text{ m/s}^2$ y desviación estándar de $7,11 \text{ m/s}^2$, mientras que los datos propios en modo de aceleración lineal presentan mediana $\approx 2,4 \text{ m/s}^2$ y $\text{std} \approx 4,1 \text{ m/s}^2$. Esta discrepancia no corresponde al modo de aceleración total ($\sim 9,8 \text{ m/s}^2$ estático, que incluye gravedad) ni al lineal ($\sim 0 \text{ m/s}^2$ estático, ya descontada); sugiere una compensación de gravedad parcial o no documentada en Bike&Safe. El impacto en el modelo es directo: mezclar ambas fuentes hace que el StandardScaler compute medias y desviaciones que no representan bien a ninguna de las dos, introduciendo ruido estructural que desestabiliza el entrenamiento. Por esta razón se decidió utilizar exclusivamente los datos propios.

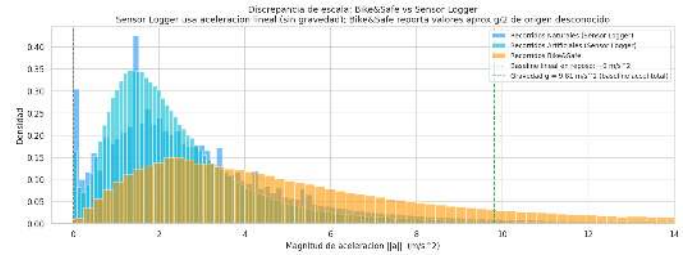


Fig. 8. Distribución de la magnitud de la aceleración para los distintos conjuntos de datos. Se evidencia una clara diferencia entre el dataset Bike&Safe y los datos propios, el cual no es explicado de forma clara por los umbrales de reposo y gravedad estándar.

2. Preprocesamiento de las series IMU y GPS

El script *combine_recordings.py* toma cada carpeta de grabación de Sensor Logger y produce un único CSV por recorrido mediante las siguientes etapas:

- Armonización de frecuencia. Los sensores se remuestrean a 50 Hz mediante promediado en bins de 20 ms, reduciendo el ruido de alta frecuencia y unificando la tasa de todas las fuentes.
- Interpolación IMU. Los valores faltantes en los canales IMU se cubren con forward-fill + backward-fill seguidos de interpolación lineal, garantizando series continuas sin NaN.
- Relleno GPS. Las coordenadas GPS, disponibles únicamente a $\sim 1 \text{ Hz}$, se propagan hacia adelante (forward-fill) para que todas las filas del CSV tengan una posición asociada, lo que facilita la asignación geoespacial por ventana.

3. Preprocesamiento de anotaciones

Para mitigar el retardo humano y los toques accidentales generados en los recorridos con eventos críticos de seguridad artificiales, se aplican automáticamente dos pasos sobre las anotaciones de cada grabación:

- Deduplicación. Las anotaciones separadas por menos de 2 s se agrupan y se conserva únicamente la última de cada cadena, asumiendo que corresponde al toque intencionado del ciclista.
- Corrección de instante. Cada etiqueta se desplaza hacia atrás al instante del pico sensorial más próximo que precede la pulsación, buscando en una ventana de entre 0,3 s y 4 s antes del tiempo registrado por la app. El canal de búsqueda depende del tipo de evento: valor absoluto del eje Z del acelerómetro para baches, magnitud total del acelerómetro para frenadas, y valor absoluto del eje Z del giroscopio para esquivadas. El pico se acepta solo si supera $1,5\times$ la mediana de la ventana; en caso contrario se aplica un desplazamiento fijo de respaldo de 2 s. Esto alinea la etiqueta con el inicio real del evento en la señal IMU, mejorando la coherencia entre anotación y fragmento de entrenamiento.

4. Análisis Exploratorio

El conjunto de datos unificado comprende 2.760.216 muestras IMU a 50 Hz, distribuidas en dos fuentes. La fuente natural aporta 2.065.293 muestras de 37 recorridos reales en Bogotá, y la fuente artificial aporta 694 923 muestras de 11 recorridos diseñados para provocar CSE, con 335 anotaciones manuales en tres categorías: 152 baches, 96 esquivadas y 87 frenadas de emergencia. No se presentan valores faltantes en los seis canales IMU tras el preprocesamiento.

Las estadísticas descriptivas muestran medias cercanas a cero en todos los canales, consistente con el uso de aceleración lineal sin componente gravitacional. Las magnitudes de desviación estándar son comparables entre fuentes, del orden de 1,1–1,3 m/s² para el eje X y de 2,0–2,6 m/s² para el eje Y, aunque los valores extremos son ligeramente mayores en los recorridos naturales, lo que refleja la mayor variedad de condiciones de vía.

La Figura 1 muestra la distribución de densidad de cada canal IMU por fuente, donde ambas poblaciones se superponen ampliamente, confirmando que los datos son comparables en escala. La Figura 2 muestra la distribución de magnitudes de aceleración y giroscopio; los perfiles de ambas fuentes son similares, lo que valida la estrategia de mezclarlas en el entrenamiento heurístico. La Figura 3 muestra las matrices de correlación entre canales.

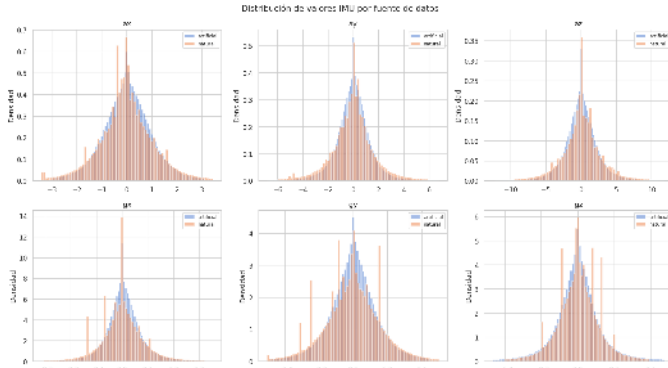


Fig. 9. Comparación de los valores IMU por fuente de datos

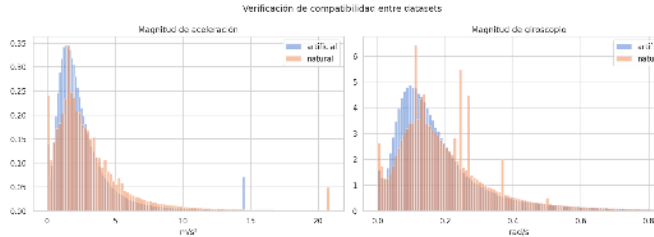


Fig. 10. Comparación de valores de aceleración y magnitud del giroscopio por fuente de datos

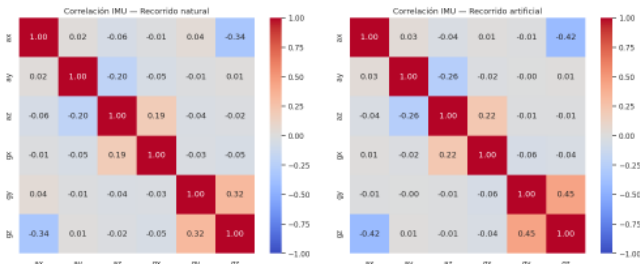


Fig. 11. Matrices de correlación por fuente de datos

Las matrices de correlación revelan dependencias significativas que responden a la configuración mecánica del dispositivo en la bicicleta. Contrario a una independencia total, se observa una correlación negativa moderada entre ax y gz (−0.34 en el recorrido natural y −0.42 en el artificial). Este fenómeno es físicamente consistente con la dinámica de viraje: al producirse una rotación en el eje vertical del manillar (gz), se genera una aceleración lateral centrípeta (ax). El signo negativo es un subproducto del sistema de coordenadas del sensor, indicando que el aumento en la velocidad de giro se traduce en una aceleración proporcional en sentido opuesto en el eje lateral del smartphone.

Por otro lado, la correlación entre los ejes ay y az (−0.20 y −0.26 respectivamente) refleja el efecto del montaje inclinado a 40°. Debido

a esta angulación, los vectores de fuerza provenientes de impactos verticales y la misma componente gravitatoria se proyectan de forma simultánea en ambos ejes. La correlación negativa sugiere que, bajo esta geometría específica, las variaciones de inclinación del cuadro durante el pedaleo provocan que el vector de gravedad se desplace entre un eje y otro de manera inversa.

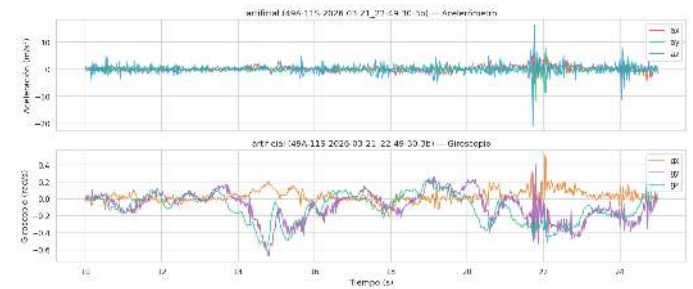
Finalmente, se destaca que las correlaciones son sistemáticamente más altas en el recorrido artificial. Esto se debe a que las maniobras controladas y repetitivas acentúan la firma física del movimiento, reduciendo el ruido estocástico presente en el entorno natural y validando la capacidad del dataset para capturar patrones cinemáticos consistentes.

5. Ventanas temporales

La segmentación de las series temporales se realizó mediante ventanas deslizantes de longitud fija aplicadas independientemente sobre cada recorrido y vuelta. Cada ventana abarca 128 muestras consecutivas a 50 Hz, equivalentes a 2,56 segundos de señal, con un desplazamiento (stride) de 64 muestras que produce un solapamiento del 50 % entre ventanas adyacentes. Este tamaño fue seleccionado por ser suficiente para capturar la duración típica de un CSE completo, que en la literatura de HAR se estima entre 1 y 3 segundos [4], y está alineado con el parámetro empleado en trabajos similares sobre señales IMU de ciclistas [7].

El proceso produjo un total de 43.058 ventanas antes de la aumentación: 32.218 provenientes de recorridos naturales y 10.840 de recorridos artificiales, de las cuales 668 contienen al menos una anotación de CSE. Cada ventana anotada fue expandida en 3 copias sintéticas, generando 2.004 ventanas adicionales para un total de 45.062 ventanas post-aumentación, con clases de CSE balanceadas a 912 (bache), 576 (esquivada) y 516 (freno de emergencia).

Para compensar el severo desbalance entre la clase normal y las clases de eventos (relación aproximada de 12:1), se aplicó data augmentation exclusivamente sobre las ventanas anotadas del conjunto de entrenamiento supervisado. Cada ventana de evento original fue expandida en 3 copias sintéticas mediante tres transformaciones compatibles con señales físicas: jitter gaussiano ($\sigma = 5\%$ de la desviación estándar por canal), escalado uniforme aleatorio en el rango [0,90 – 1,10], e inversión temporal de la ventana. El resultado final fue de 43.058 ventanas totales para el conjunto de entrenamiento, con las clases de CSE balanceadas a 1.302 (bache), 868 (esquivada) y 805 (freno) ventanas respectivamente.



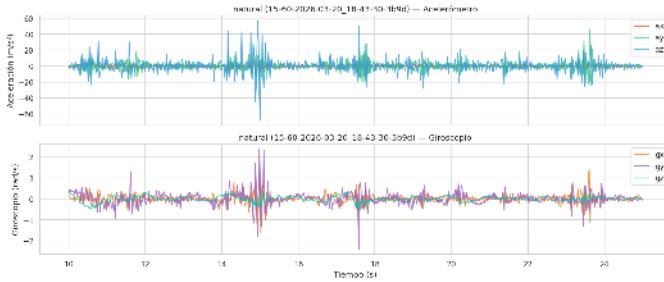


Fig. 12. Ejemplo de series temporales

6. Feature Engineering

Por cada una de las seis señales inerciales (aceleración en los ejes x, y, z y velocidad angular en los ejes x, y, z) se extraen diez características estadísticas por ventana: media, desviación estándar, mínimo, máximo, rango, RMS (energía), asimetría (skewness), kurtosis, percentil 5 y percentil 95. Adicionalmente, se calculan una parte de esas estadísticas (6 de 10) sobre la magnitud de aceleración y la magnitud de giroscopio, derivadas como la norma euclidiana de los respectivos ejes. El vector de entrada resultante al MLP tiene por tanto una dimensionalidad fija independiente de la longitud de la ventana, que corresponde a 72 componentes o features. En la figura 5 se puede observar la distribución de las características seleccionadas.

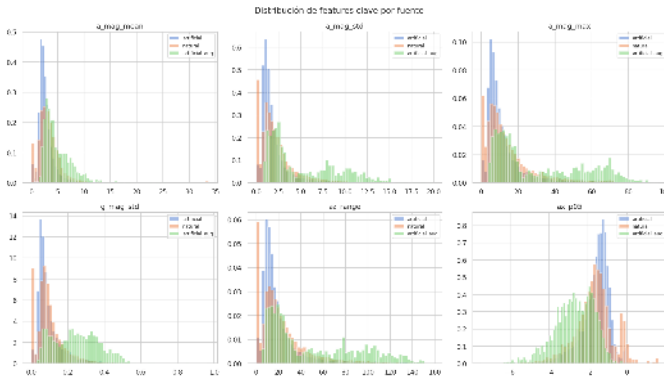


Fig. 13. Distribución de las características seleccionadas

7. Clasificación de clases

Para el Modelo A (heurístico, 3 clases), las ventanas son etiquetadas automáticamente a partir de dos puntuaciones computadas sobre el vector de características de cada ventana. El severity score combina cuatro descriptores sensibles a maniobras bruscas: el percentil 5 del eje X de aceleración, el percentil 99 de la magnitud de aceleración, el percentil 99 de la magnitud de giroscopio, y la desviación estándar de la magnitud de giroscopio. El bump score captura impactos verticales mediante el valor máximo absoluto del eje Z.

Los umbrales de clasificación se calibraron sobre el conjunto de entrenamiento fijando tasas objetivo: el 8 % de las ventanas con mayor severity score se etiquetan como severo (umbral = 28,44) y el 18 % de las ventanas con mayor bump score se etiquetan como bache (umbral = 19,18), de forma que ambas reglas son mutuamente excluyentes con precedencia de bache. El resto se clasifica como normal. La distribución resultante en el conjunto global es de 36.595 ventanas normales (81,21 %), 4.317 severo (9,58 %) y 4.150 bache (9,21 %), como se muestra en la Figura 14.

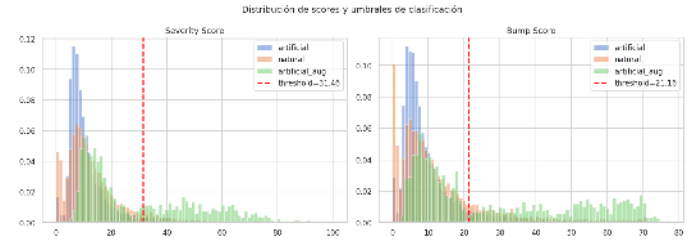


Fig. 14. Distribución de umbrales de clasificación

IX. ENFOQUE PROPUESTO BASADO EN REDES NEURONALES SUPERFICIALES

Descripción General

Se propone un enfoque basado en un Perceptrón Multicapa (MLP) para la clasificación multiclase de CSE, utilizando datos inerciales multivariados provenientes de sensores de smartphone (acelerómetro y giroscopio). A diferencia de la red convolucional profunda, que opera directamente sobre las ventanas crudas de señal, el MLP recibe como entrada un vector de características estadísticas extraídas de cada ventana temporal mediante técnicas de Feature Engineering. Este enfoque reduce la dimensionalidad del problema antes de la etapa de clasificación, siguiendo la metodología ampliamente documentada en la literatura de HAR [5]. El modelo opera sobre ventanas segmentadas de 128 muestras a 50 Hz, equivalentes a aproximadamente 2,56 segundos de recorrido, con un solapamiento del 50% entre ventanas consecutivas (stride de 64 muestras). Este tamaño de ventana fue seleccionado por ser suficiente para capturar la duración típica de un evento completo, como un frenado brusco o un impacto por bache.

Arquitectura MLP Propuesta

La entrada del MLP corresponde a un vector de 72 características estadísticas por ventana, calculadas a partir de los seis canales IMU y sus magnitudes. Estas features resumen propiedades como media, desviación estándar, percentiles, energía, máximos, mínimos y medidas de variabilidad, reduciendo cada ventana temporal de 128 x 6 valores crudos a una representación tabular compacta.

La arquitectura implementada consta de dos capas ocultas con activación ReLU y regularización mediante Dropout, detallado en la Figura 15.

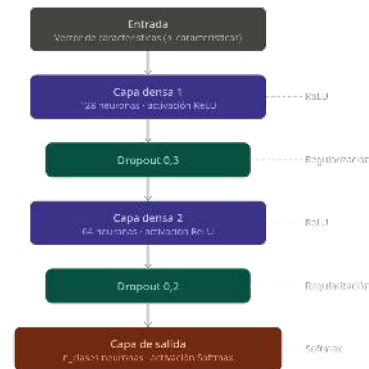


Fig. 15. Flujo de la arquitectura de MLP propuesta

La primera capa oculta (128 neuronas) permite modelar relaciones no lineales entre las características estadísticas, mientras que la segunda (64 neuronas) actúa como etapa de compresión antes de la clasificación final. Las capas de Dropout mitigan el sobreajuste dado el tamaño moderado del conjunto de entrenamiento. La capa de salida emplea activación Softmax, produciendo una distribución de

probabilidad sobre las clases. El modelo se entrena con el optimizador Adam (con una tasa de aprendizaje de 1×10^{-3}), función de pérdida categorica cross entropy, y pesos de clase balanceados calculados inversamente proporcionales a la frecuencia de cada clase en el conjunto de entrenamiento.

El modelo se evalúa en dos configuraciones: una variante heurística de tres clases (normal, bache, severo), cuyas etiquetas se derivan automáticamente a partir de umbrales calibrados sobre los scores de severidad e impacto vertical, y una variante supervisada de cuatro clases (normal, bache, esquivada, freno) que incorpora las anotaciones manuales recolectadas con datos propios. Esta dualidad permite contrastar el desempeño del sistema tanto en escenarios donde no se dispone de etiquetas manuales como en condiciones de supervisión plena.

X. ENFOQUE PROPUESTO BASADO EN REDES NEURONALES PROFUNDAS

Descripción General

A diferencia del MLP, que recibe un vector de características estadísticas previamente calculadas, la CNN 1D opera directamente sobre las ventanas crudas de señal inercial. Cada muestra de entrada corresponde a una matriz temporal de 128 muestras por 6 canales, compuesta por las tres componentes del acelerómetro y las tres componentes del giroscopio. Esta representación permite que la red aprenda automáticamente filtros temporales capaces de detectar patrones locales asociados a impactos, oscilaciones laterales y desaceleraciones abruptas.

Configuración de entrenamiento

La CNN utiliza las mismas particiones de entrenamiento, validación y test que el MLP para asegurar una comparación directa. Las señales se estandarizan por canal usando estadísticas calculadas únicamente sobre el conjunto de entrenamiento. En el caso supervisado, se aplica SMOTE sobre las ventanas aplanadas y luego se reconstruye la forma tridimensional para balancear las clases minoritarias. El modelo se entrena con Adam, sparse categorical crossentropy, accuracy como métrica de entrenamiento, reducción adaptativa de la tasa de aprendizaje y evaluación final mediante accuracy y F1 macro.

El modelo se evalúa en dos configuraciones: una variante heurística de tres clases (normal, bache, severo), cuyas etiquetas se derivan automáticamente a partir de umbrales calibrados sobre los scores de severidad e impacto vertical, y una variante supervisada de cuatro clases (normal, bache, esquivada, freno) que incorpora las anotaciones manuales recolectadas con datos propios. Esta dualidad permite contrastar el desempeño del sistema tanto en escenarios donde no se dispone de etiquetas manuales como en condiciones de supervisión plena.

Arquitectura de CNN propuesta

La arquitectura de la CNN se encuentra descrita en la figura. La entrada a la CNN consiste en una matriz temporal de 128 muestras por 6 canales, que representa una ventana cruda ya descrita anteriormente. Esta entrada pasa por una secuencia de tres capas convolucionales, todas con funciones de activación ReLU, batch normalization y dropout. El tamaño de los kernels es de 5, 3 y 3 respectivamente, y el stride es igual a 1 en todos los casos. Esto significa que los kernels son de dimensiones (5, 6), (3, 6) y (3, 6) respectivamente. Esto permite que la red neuronal aprenda a extraer características que no solo se desenvuelven en el tiempo sino que también implican la interacción de los distintos canales (por ejemplo,

un viraje implica tanto mediciones elevadas en un componente del giroscopio como del acelerómetro). La primera capa convolucional utiliza 64 filtros, la siguiente 128 y la última 256. De esta forma la red puede extraer cada vez más características de mayor nivel de la señal.

Se utilizó max pooling para las primeras dos capas y global average pooling para la última capa, pues permite una mejor generalización que un simple aplanamiento. Por último, se utilizan dos capas densas para la clasificación, una de 128 neuronas para modelar unas relaciones no lineales iniciales a partir de las características extraídas por la CNN, y una última de clasificación que utiliza una función softmax.



Fig. 16. Flujo de la arquitectura de CNN propuesta

XI. RESULTADOS EXPERIMENTALES: COMPARACIÓN ENTRE MLP Y CNN 1D

Configuración experimental

La división experimental quedó definida por máscaras de entrenamiento, validación y test guardadas en los artefactos del notebook. Se realizó un split de 70/15/15 para entrenamiento, validación y test respectivamente. Para el enfoque heurístico se usaron 30.140 ventanas de entrenamiento, 6.459 de validación y 6.459 de test. Para el enfoque supervisado se usaron 9.591 ventanas crudas de entrenamiento antes de SMOTE, 1.626 de validación y 1.627 de test. En la CNN supervisada, SMOTE expandió el conjunto de entrenamiento a 28.480 ventanas balanceadas. Todos los modelos

fueron evaluados sobre test oculto para reportar las métricas finales comparables.

Ambos modelos se entrenaron con el optimizador Adam durante un máximo de 100 épocas con reducción adaptativa de la tasa de aprendizaje (ReduceLROnPlateau), convergiendo a las 57 épocas (Modelo A) y 86 épocas (Modelo B).

Desempeño en validación

La Tabla 1 resume las métricas finales obtenidas por las cuatro configuraciones evaluadas en el notebook: MLP heurístico, MLP supervisado, CNN 1D heurística y CNN 1D supervisada.

Modelo	Modelo	Entrada	Accuracy	F1 Macro
Heurístico (3 clases)	MLP	Features estadísticas	0,9866	0,9687
Heurístico (3 clases)	CNN 1D	Señal cruda 128 x 6	0,9578	0,8989
Heurístico (3 clases)	CNN 1D (AutoML)	Señal cruda 128 x 6	0,9768	0,9426
Supervisado (4 clases)	MLP	Features estadísticas	0,9736	0,8738
Supervisado (4 clases)	CNN 1D	Señal cruda 128 x6	0,9883	0,9312
Supervisado (4 clases)	CNN 1D (AutoML)	Señal cruda 128 x6	0,9680	0,7890

Tabla 3. Comparación final de Accuracy y F1 Macro en test oculto

En el enfoque heurístico, la red shallow MLP obtiene el mejor desempeño, con un *accuracy* de 0,9866 y un F1 macro de 0,9687, mientras que las arquitecturas convolucionales logran un F1 macro máximo de 0,9426 (CNN AutoML). Este resultado es consistente con el origen matemático de las pseudoetiquetas: al haber sido generadas a partir de scores construidos mediante características estadísticas y umbrales rígidos, el MLP se ve directamente beneficiado, pues ingiere exactamente el mismo tipo de información numérica pre-procesada.

Por el contrario, en el enfoque supervisado (anotaciones humanas de eventos reales), la superioridad se invierte hacia el Deep Learning. Evaluada en el conjunto de prueba oculto bajo una estricta separación por rutas para evitar fuga de datos temporal, la CNN 1D de arquitectura manual alcanza un sobresaliente F1 macro de 0,9312 (*accuracy* = 0,9883), superando significativamente al MLP supervisado (F1 macro = 0,8738). Resulta notable que el algoritmo de optimización bayesiana (AutoML KerasTuner) propuso una arquitectura convolucional más profunda que, si bien maximizó el *accuracy* de validación, sufrió una severa penalización en su F1 macro durante la prueba oculta (0,7890). Este fenómeno evidencia que optimizar modelos inerciales fuertemente desbalanceados priorizando métricas globales como el *accuracy* conduce al

sobreajuste hacia la clase mayoritaria. Por tanto, el diseño empírico de la CNN 1D manual demostró poseer el campo receptivo temporal idóneo para generalizar y descubrir las firmas cinemáticas complejas que diferencian baches, esquivadas y frenadas de emergencia en entornos urbanos no vistos.

Para ilustrar este comportamiento, las siguientes figuras detallan el rendimiento de los modelos en el Test Set oculto. Se presentan las matrices de confusión consolidadas del MLP y se contrastan con las matrices de la CNN supervisada manual, donde se evidencia la reducción de falsos positivos gracias a la convolución sobre el eje del tiempo. Adicionalmente, se incluyen las gráficas comparativas de *accuracy* y F1 macro, las cuales resumen visualmente el impacto de preservar la estructura espacial de las señales para el reconocimiento de actividades anómalas en el ciclismo urbano.

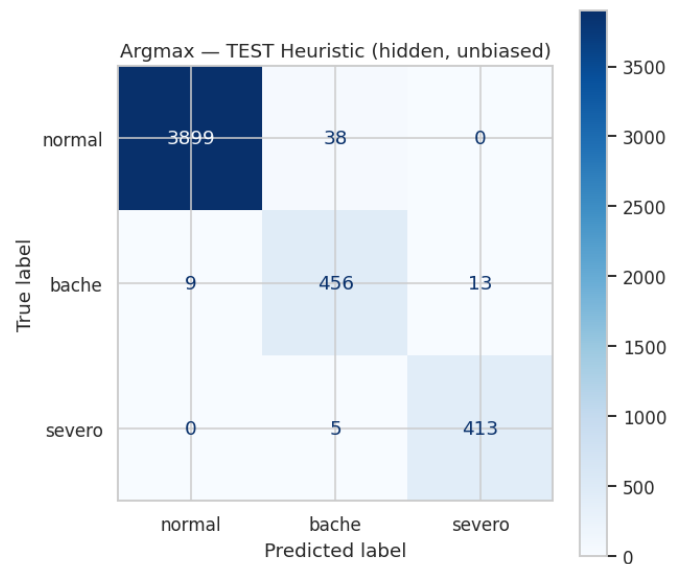


Fig. 17. Matriz de confusión del modelo MLP Heurístico

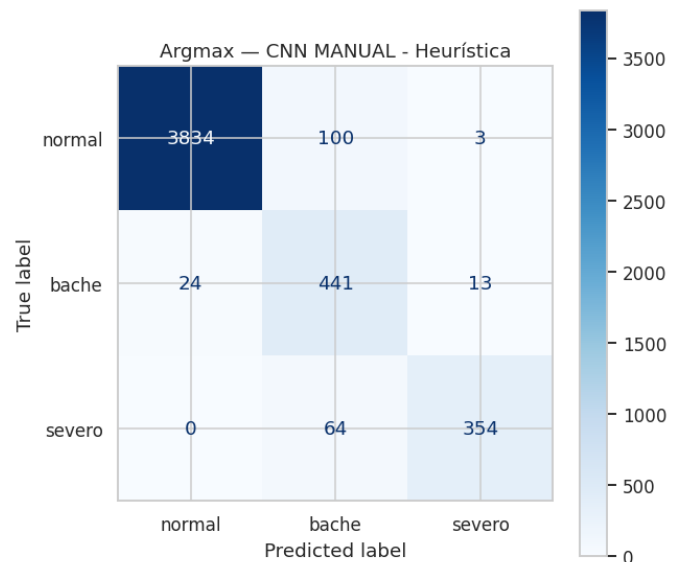


Fig. 18. Matriz de confusión del modelo CNN Heurístico

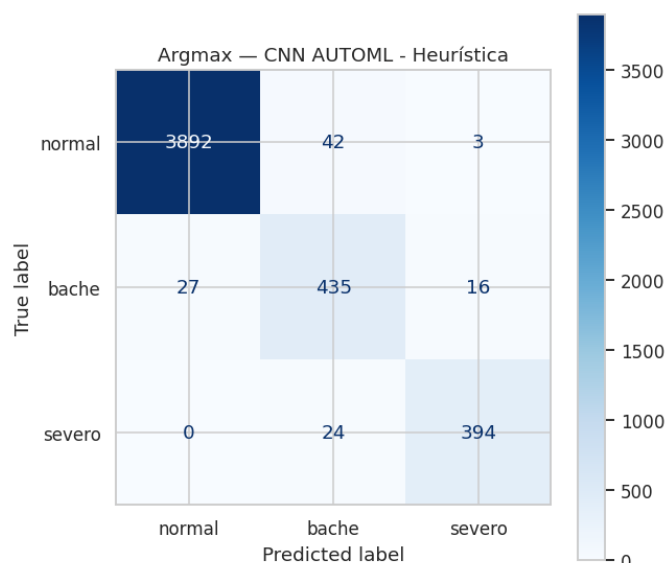


Fig. 19. Matriz de confusión del modelo CNN (AutoML) Heurístico

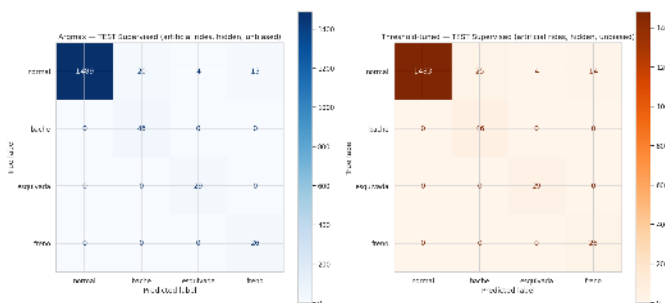


Fig. 20. Matrices de confusión para el modelo MLP Supervisado

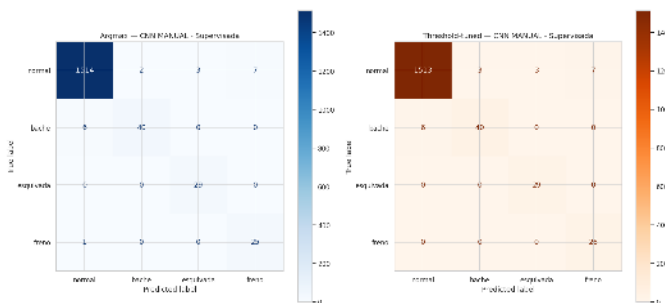


Fig. 21. Matrices de confusión para el modelo CNN Supervisado

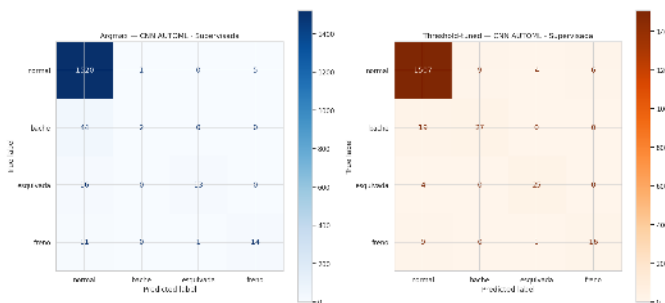


Fig. 22. Matrices de confusión para el modelo CNN (AutoML) Supervisado

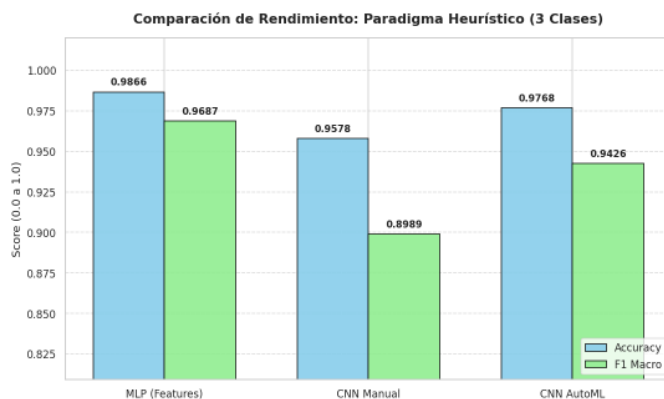


Fig. 23. Comparación de rendimiento paradigma heurístico

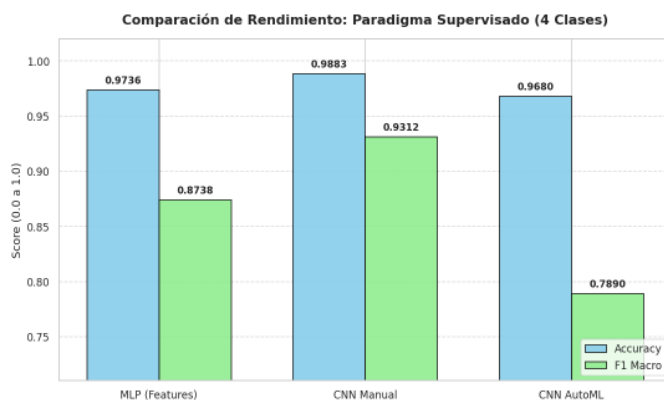


Fig. 24. Comparación de rendimiento paradigma supervisado

Verificación del sobreajuste

Al observar las curvas de aprendizaje, se evidencia que los modelos aprenden correctamente sin evidencia de sobreajuste. También se aprecia un fenómeno característico: en diversas épocas, la métrica de validación supera ligeramente al conjunto de entrenamiento. Este comportamiento, lejos de ser una anomalía, es una evidencia empírica de la eficacia de las técnicas de regularización implementadas (específicamente Dropout y Batch Normalization).

Durante la fase de entrenamiento, la arquitectura opera bajo una configuración restrictiva, donde la desactivación aleatoria de neuronas (Dropout) y la normalización basada en batches limitan la capacidad de aprendizaje total del modelo, forzándolo a aprender características robustas y no solo a memorizar los datos. Por el contrario, durante la validación, el modelo utiliza su arquitectura completa y los parámetros estadísticos normalizados acumulados, logrando un desempeño superior al no verse limitado por los mecanismos de regularización activa.

Adicionalmente, dado que el conjunto de entrenamiento incluye técnicas de aumento de datos (data augmentation), el modelo se enfrenta a una tarea más compleja durante el aprendizaje en comparación con el conjunto de validación, el cual preserva la naturaleza inalterada de las señales inerciales. Esta brecha en el desempeño reafirma que los modelos, tanto manuales como optimizados por AutoML, alcanzan un estado de convergencia adecuado, evitando el sobreajuste (overfitting) y manteniendo una alta capacidad de generalización ante datos no vistos, como se confirma en la estabilidad observada en el conjunto de prueba oculto.

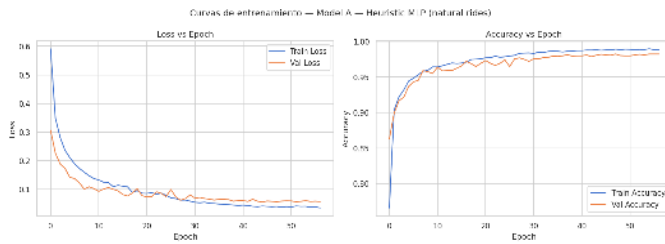


Fig 25. Curvas de entrenamiento del modelo MLP Heurístico

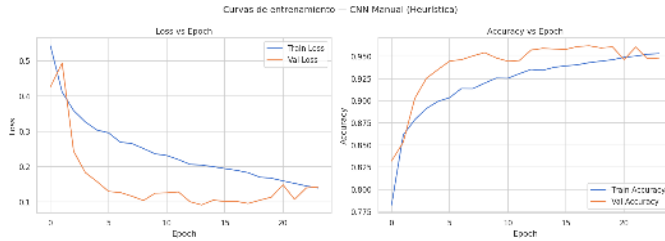


Fig 26. Curvas de entrenamiento del modelo CNN Heurístico

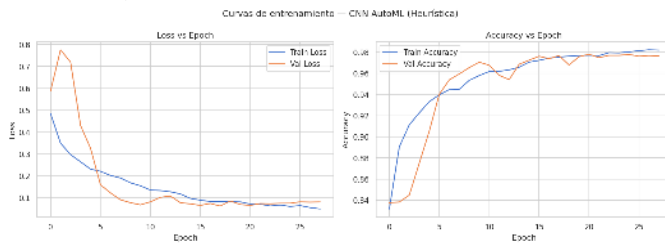


Fig 27. Curvas de entrenamiento del modelo CNN (AutoML) Heurístico

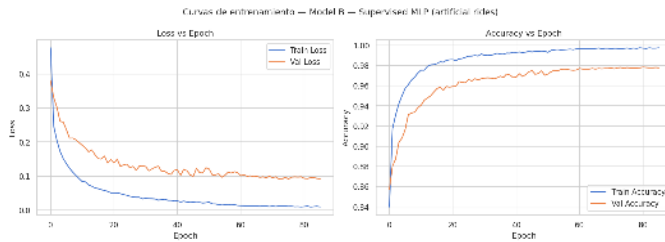


Fig 28. Curvas de entrenamiento del modelo MLP Supervisado

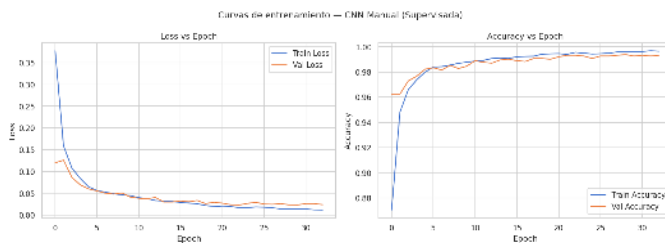


Fig 29. Curvas de entrenamiento del modelo CNN Supervisado

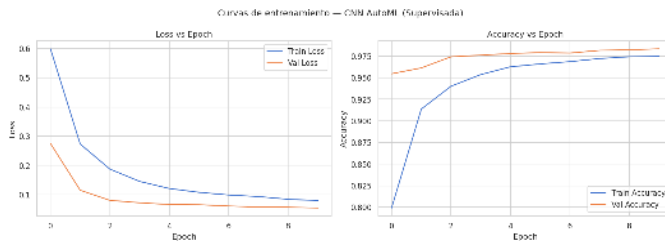


Fig 30. Curvas de entrenamiento del modelo CNN (AutoML) Supervisado

XII. CREACIÓN DE NODO IoT

1. Lista de Materiales

Con el objetivo de materializar la arquitectura IoT propuesta con un hardware de bajo costo, especializado y eficiente se adquirieron los siguientes materiales, centrados en una arquitectura basada en un ESP-32

- **Unidad de Procesamiento:** Microcontrolador ESP32 (NodeMCU DevKitC). Seleccionado por su arquitectura de doble núcleo a 240 MHz, 520 KB de RAM interna y conectividad Wi-Fi/Bluetooth integrada.
- **Unidad de Medición Inercial (IMU):** Sensor MPU-6050. Integra un acelerómetro de 3 ejes y un giroscopio de 3 ejes con un procesador digital de movimiento (DMP), comunicado por bus I2C.
- **Unidad de Posicionamiento:** Módulo GPS NEO-6M con antena pasiva cerámica integrada. Comunicación vía UART a una tasa de transferencia nativa de 9600 bps. Captura posiciones a 1Hz.
- **Unidad de Almacenamiento Local:** Módulo Lector de Tarjetas MicroSD con interfaz SPI y regulador de nivel lógico integrado, junto a una tarjeta MicroSD de 32 GB (Class 10) formateada en sistema de archivos FAT32.
- **Componentes de Soporte:** Tarjeta perforada de baquelita (Perfboard), tarjeta de pruebas (Protoboard), zócalos para integrados (bornas hembra), cables jumper de conexión, interruptor de palanca, cable de programación micro-USB y un banco de energía portátil de 5V como fuente de alimentación regulada.



Fig 31. Fotografía de los materiales empleados para la construcción del nodo IoT

2. Fase de Protoboard y Pruebas Iniciales

La primera etapa consistió en el mapeo lógico y la validación eléctrica de las interconexiones en una placa de pruebas (Protoboard). El diseño se estructuró dividiendo los periféricos según los buses de comunicación del ESP32:

1. Bus I2C (Telemetría inercial): El MPU-6050 se conectó a los pines nativos GPIO 21 (SDA) y GPIO 22 (SCL). Se energizó a 3.3V utilizando líneas cortas para mitigar el ruido electromagnético.
2. Bus SPI (Caja negra / SD): El módulo MicroSD se enlazó mediante el bus VSPI del ESP32, asignando los pines GPIO 23 (MOSI), GPIO 19 (MISO), GPIO 18 (SCK) y

GPIO 5 como Selector de Chip (CS). Debido al consumo pico en los procesos de escritura magnética, se alimentó directamente desde la línea de 5V.

3. Bus UART2 (Geolocalización): El módulo GPS NEO-6M se vinculó cruzando los canales de transmisión hacia los pines GPIO 16 (RX2) y GPIO 17 (TX2) del microcontrolador.

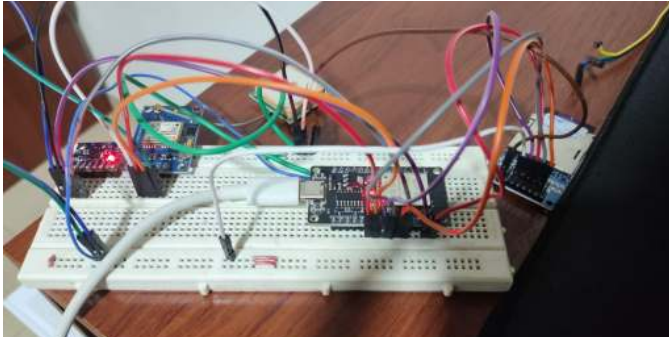


Fig 32. Fotografía de la construcción inicial del circuito con una protoboard

Con el circuito cableado en la protoboard, se desarrollaron en arduino IDE scripts de diagnóstico unitario para asegurar la salud de los componentes:

- Auditoría del Bus I2C: Se ejecutó un escáner de direcciones para confirmar la presencia del MPU-6050 en la dirección hexadecimal estándar 0x68 a una velocidad de bus configurada a 400 kHz (Fast Mode).
- Test de Estrés de la SD: Se cargó un algoritmo de escritura cíclica que abría, modificaba y cerraba un archivo .csv cada 20 ms, verificando que no existieran pérdidas de paquetes (buffer overflows) ni bloqueos por caídas de tensión en el bus SPI.
- Enganche de Satélites en Frío (Cold Start): El módulo GPS se dispuso en un espacio abierto para medir el tiempo de adquisición del primer mapa de satélites (Time To First Fix - TTFF). Se comprobó visualmente que la transición del estado de búsqueda al estado de enganche se diera al activarse el parpadeo constante del LED indicador físico del módulo.
- Monitoreo general de sensores: Se realizaron pruebas en las que se obtenían los valores IMU inerciales, la ubicación GPS, imprimiéndose en el monitor serial cada uno de estos valores. Como proceso final, estos valores se grababan en un archivo .csv de la tarjeta microSD, a fin de realizar una comprobación final

```

pruebaAcelerometro.ino
46 void loop() {
47   // A. Alimentar el decodificador de GPS
48   while (Serial2.available() > 0) {
49     gps.encode(Serial2.read());
50   }
51
52   // B. Leer métricas del MPU
53   sensors_event_t a, g, temp;
54   mpu.getEvent(&a, &g, &temp);
55
56   // Aplicar calibración al eje Z
57   float accZ_calibrada = a.acceleration.Z - OFFSET_Z;
58
59   // C. Crear la línea de datos en formato string (Estilo CSV)
60   String lineaDatos = "";
61   lineaDatos += "AccX:" + String(a.acceleration.x, 2) + ",";
62   lineaDatos += "AccY:" + String(a.acceleration.y, 2) + ",";
63   lineaDatos += "AccZ:" + String(accZ_calibrada, 2) + ",";
64   lineaDatos += "GiroZ:" + String(g.gyro.z, 2) + ",";
65
66   if (gps.location.isInvalid()) {
67     lineaDatos += "Lat:" + String(gps.location.lat(), 8) + ",";
68     lineaDatos += "Lng:" + String(gps.location.lng(), 8) + ",";
69     lineaDatos += "Sats:" + String(gps.satellites.value());
70   } else {
71     lineaDatos += "Lat:0.0,Lng:0.0,Sats:" + String(gps.satellites.value()) + " (Buscando)";
72   }
73 }
  
```

Output: Serial Monitor: X

Message (Error to send message to 'ESP32 Dev Module' on '/dev/ttyUSB0')

21:54:03.312 -> AccX:0.45, AccY:0.16, AccZ:0.87, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:03.594 -> AccX:0.45, AccY:0.18, AccZ:0.84, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:03.728 -> AccX:0.43, AccY:0.23, AccZ:0.88, GiroZ:0.02, Lat:0.000000, Lng:0.000000, Sats:4

21:54:03.521 -> AccX:0.44, AccY:0.21, AccZ:0.90, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:04.113 -> AccX:0.44, AccY:0.20, AccZ:0.92, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:04.539 -> AccX:0.44, AccY:0.21, AccZ:0.90, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:04.592 -> AccX:0.45, AccY:0.19, AccZ:0.86, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:04.724 -> AccX:0.44, AccY:0.20, AccZ:0.88, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:04.951 -> AccX:0.45, AccY:0.21, AccZ:0.85, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:05.143 -> AccX:0.43, AccY:0.18, AccZ:0.86, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:05.335 -> AccX:0.41, AccY:0.18, AccZ:0.84, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:05.588 -> AccX:0.43, AccY:0.19, AccZ:0.86, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:05.782 -> AccX:0.41, AccY:0.23, AccZ:0.93, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:05.976 -> AccX:0.42, AccY:0.26, AccZ:0.96, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:06.168 -> AccX:0.39, AccY:0.18, AccZ:0.95, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

21:54:06.362 -> AccX:0.38, AccY:0.18, AccZ:0.97, GiroZ:0.03, Lat:0.000000, Lng:0.000000, Sats:4

Fig 33. Screenshot de las pruebas realizadas en arduino IDE

3. Ensamblaje Final en Perfboard (Soldadura)

Las pruebas en protoboard demostraron que los falsos contactos mecánicos causados por el movimiento arruinaban las lecturas de los sensores. Para garantizar la rigidez y viabilidad técnica en las calles de Bogotá, el circuito se migró de forma definitiva a una placa perforada (Perfboard).

Se soldaron zócalos hembra en la baquelita para permitir la remoción de los componentes principales en caso de falla y se distribuyeron pistas de cableado eléctrico para los rieles comunes de potencia (GND y 5V). El nodo final se encapsuló en un estuche plástico diseñado para fijarse mecánicamente al cuadro de la bicicleta en un ángulo aproximado de 40° con respecto a la horizontal. Se tomaron medidas adicionales ante las posibles vibraciones y movimientos propios de los recorridos ciclistas como puntos de soldadura robustos, piezas plásticas de apoyo y sujeción extra con silicona caliente.



Fig 34. Fotografía del montaje final protegido con la montura ciclista



Fig 35. Fotografía del circuito final descubierto

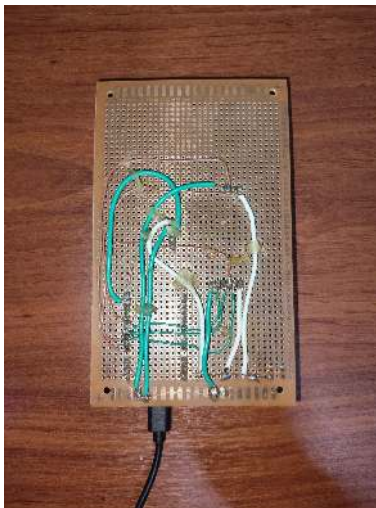


Fig 36. Fotografía de soldadura trasera

4. Optimización, Cuantización y Portabilidad del Modelo (Pipeline TinyML)

El despliegue de modelos de Deep Learning en sistemas embebidos de recursos restringidos (Edge Computing) exige una fase rigurosa de optimización. Una arquitectura convolucional nativa en Python opera bajo una precisión de punto flotante de 32 bits, lo que genera un tamaño de binario y un consumo de memoria RAM prohibitivos para los microcontroladores comerciales. Para transformar la red convolucional supervisada CNN en un motor ejecutable en el borde (Edge AI), se implementó un pipeline de optimización basado en el framework TensorFlow Lite Micro, estructurado en cuatro etapas consecutivas: cuantización, serialización, conversión a código C++ y alineación del pipeline inercial.

El proceso de compresión y portabilidad de la red neuronal convolucional CNN al microcontrolador se logró mediante Cuantización Post-Entrenamiento de enteros puros (*Int*). Partiendo del modelo nativo de Keras, se utilizó un generador de datos representativos con ventanas inerciales reales de Bogotá para calibrar los rangos dinámicos del chip, transformando los pesos y activaciones de punto flotante *Float32* a enteros discretos de 8 bits *Int8*. El grafo optimizado se serializó en formato plano FlatBuffers (.tflite) y se convirtió en un arreglo hexadecimal estático de C++ (modelo_esp32.h) forzando una alineación de 32 bits para la arquitectura de la CPU. Para garantizar la fidelidad matemática en el borde, el firmware del ESP32 incorporó un pipeline que calcula el Z-Score de los sensores inerciales en cada tick de reloj y aplica una función lambda de cuantización al vuelo, mapeando el reposo físico directamente al Zero Point (3) y la escala (0.129156) que el modelo comprimido exige. Esta optimización TinyML permitió un despliegue sumamente eficiente en el borde, logrando una reducción crítica de recursos con un impacto casi imperceptible en la precisión predictiva, tal como se detalla en la siguiente comparativa de desempeño:

Métrica	Original (Keras Float32)	Optimizado (TFLite INT8)	Impacto / Cambio
Tamaño en Disco	2394.95 KB	227.20 KB	Reducción del 90.5%
Latencia de Inferencia	32.82 ms/ventana	0.03 ms/ventana	Aceleración 99%
Accuracy Global	0.9926	0.9920	0.06% de variación marginal
F1-Score (Macro)	0.9531	0.9487	0.45% de variación marginal

Tabla 4. Comparativa de desempeño modelo original y modelo optimizado

Al contraer el tamaño del archivo un 90.5%, el modelo se aloja cómodamente en la memoria del dispositivo sin desbordar sus particiones físicas. Asimismo, la velocidad de ejecución se aceleró un 99.9%, reduciendo el tiempo de cómputo a unos imperceptibles 0.03ms locales, ideales para la detección inmediata de anomalías en la vía. Lo más destacable del pipeline es que esta eficiencia extrema se obtuvo reteniendo casi intacta la inteligencia original del sistema,

pues la degradación del Accuracy y del F1-Score fue inferior al 0.5%, consolidando un nodo IoT de alta fidelidad, autónomo y ultra-rápido.

5. Integración Final del Firmware y Sinergia Edge-to-Cloud con ThingsBoard

El programa definitivo unifica de manera armónica el hardware diseñado y el modelo de red neuronal convolucional optimizado *INT8* en un firmware de alto rendimiento embebido en el ESP32. Operando bajo un enfoque híbrido, el sistema ejecuta de forma estrictamente síncrona la captura y el preprocesamiento de señales inerciales a una frecuencia determinista de 50 Hz (cada 20 ms), mientras gestiona en segundo plano la decodificación de tramas geoespaciales del módulo GPS NEO-6M, la escritura física en la MicroSD y el mantenimiento del socket de red. El propósito central de este software es actuar como una unidad de percepción y computación autónoma en el borde, capaz de filtrar el ruido del entorno y clasificar eventos críticos en la bicicleta en tiempo real sin depender de la nube para el procesamiento matemático.

A nivel de flujo interno, el programa ejecuta un pipeline secuencial de ingeniería de características directamente en la CPU del microcontrolador antes de alimentar la Inteligencia Artificial. En cada ciclo de reloj, el firmware extrae las lecturas del MPU-6050, remueve dinámicamente el vector de la gravedad mediante ecuaciones trigonométricas basadas en los ángulos instantáneos de pitch y roll, y estandariza los 6 canales inerciales mediante una transformación Z-Score basada en las medias y desviaciones estándar calibradas en el laboratorio. Posteriormente, una función lambda de cuantización traduce estos flotantes escalados a enteros de 8 bits vinculados a la escala y el punto cero del modelo embebido, insertando el resultado ordenadamente en una ventana deslizante de memoria.

El trigger de telemetría inteligente se activa automáticamente cada vez que la ventana deslizante se completa. En ese microsegundo, el programa invoca al intérprete de TensorFlow Lite Micro, resolviendo el grafo convolucional localmente en apenas 0.03 ms a través de un resolutor mutable de 10 operaciones de hardware. Si el algoritmo de argmax identifica una anomalía vial en la superficie de Bogotá (bache, esquivada o frenada brusca), el firmware captura las coordenadas geográficas vigentes del GPS, empaqueta los datos de confianza y georreferenciación en un payload JSON y lo publica mediante protocolo MQTT hacia el broker de ThingsBoard Cloud. Finalmente, como respaldo de seguridad y contingencia ante la pérdida de señal celular en la vía, el programa estampa en paralelo la predicción e historial inercial continuo en la tarjeta MicroSD, consolidando una "caja negra" robusta para el análisis de la siniestralidad y la infraestructura urbana.

XIII. CONSTRUCCIÓN DE MAPAS E INTEGRACIÓN CON PLATAFORMA THINGSBOARD

Mapas de calidad vial y zonas de riesgo ciclista

A partir de las predicciones del modelo supervisado usado en el pipeline de mapas y del severity score heurístico calculado sobre 37 recorridos naturales en Bogotá, se generaron tres mapas interactivos que constituyen el producto final de identificación de zonas de riesgo propuesto en los objetivos del proyecto. En esta versión, el GPS se utiliza para georreferenciar ventanas y predicciones, no como variable de entrada de los modelos MLP o CNN.

Para la construcción de los mapas, cada ventana de 2,56 s fue asociada a la coordenada GPS del instante central de la ventana. Las

coordenadas GPS del dispositivo (~1 Hz) fueron propagadas hacia adelante (forward-fill) para asegurar cobertura en todas las muestras. Con el fin de evitar que tramos recorridos múltiples veces inflaran artificialmente la intensidad del heatmap, los datos se agregaron por celdas geográficas de aproximadamente 20 m de lado, computando el severity score promedio por celda. El resultado es una representación fiel de la calidad real del pavimento, independiente de la frecuencia con que se transitó cada tramo. Se utilizó Folium para la creación de mapas en un entorno Python.

El Mapa A superpone los eventos CSE detectados por el modelo supervisado sobre la traza GPS de los recorridos. Cada marcador indica el tipo de evento: bache (naranja), esquivada (morado) y freno de emergencia (rojo). Se observa una distribución heterogénea de eventos a lo largo de las rutas, con concentraciones notables en intersecciones y zonas con cambios de superficie, coherente con las condiciones de tráfico esperadas en los corredores de las localidades de Bosa, Kennedy y Fontibón.

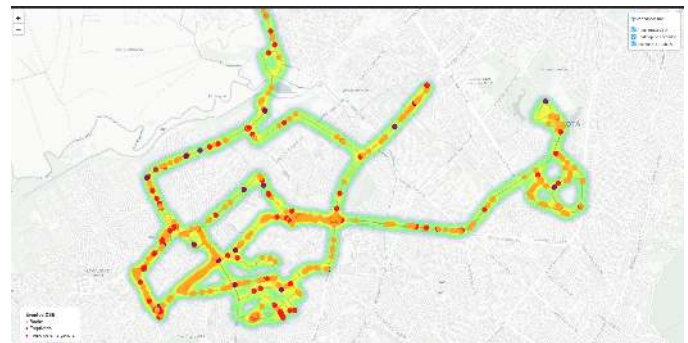


Fig 37. Mapa con eventos CSE y heatmap de severidad combinado

El Mapa B (Figura 13) muestra el heatmap de severidad promediada por celda geográfica, con gradiente continuo de azul (bajo) a rojo (alto). Se identifican zonas de alta severidad concentradas en tramos específicos, en particular en la zona suroccidental del mapa, donde convergen múltiples rutas con pavimento deteriorado. Los tramos verdes predominantes corresponden a ciclovías y vías en mejor estado. Esta distribución espacial de la calidad vial demuestra la viabilidad del sistema para producir información accionable sobre el estado de la infraestructura ciclista urbana.



Fig 38. Mapa con heatmap de severidad

Integración con plataforma ThingsBoard

Como capa de visualización operativa de la arquitectura propuesta, se configuró un dashboard interactivo en ThingsBoard Community Edition que centraliza la telemetría recibida por MQTT desde el nodo ESP32. El dashboard integra seis componentes funcionales que permiten el monitoreo en tiempo real de los eventos críticos de seguridad detectados por el modelo CNN 1D desplegado en el borde.

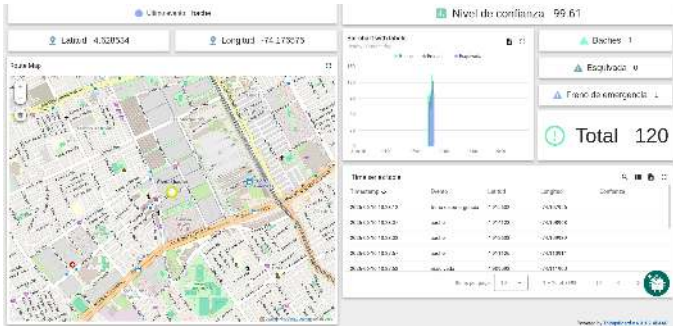


Fig 39. Dashboard mediante la plataforma ThingsBoard

El widget Route Map muestra la trayectoria GPS activa del ciclista sobre cartografía OpenStreetMap, con marcadores georeferenciados que indican la posición exacta del último evento detectado. Cada marcador incluye una etiqueta textual con el tipo de evento clasificado, permitiendo una identificación espacial inmediata de las zonas de riesgo durante el recorrido.

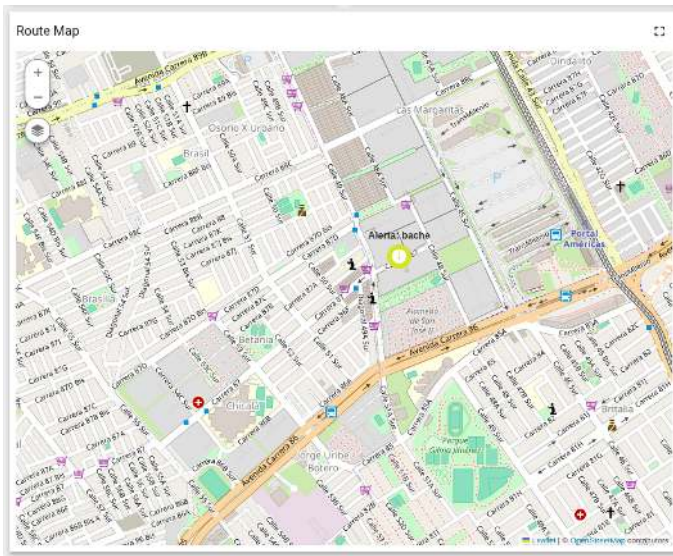


Fig 40. Widget Route Map

Los widgets de latitud y longitud exponen las coordenadas GPS del último evento registrado, actualizándose en tiempo real conforme el nodo publica nuevas alertas. El campo último evento refleja la clase predicha más reciente emitida por el clasificador embebido, mientras que el nivel de confianza muestra la probabilidad Softmax asociada a dicha predicción.



Fig 41. Widgets latitud, longitud y último evento

La gráfica de barras distribución de eventos desglosa temporalmente la frecuencia de ocurrencia de cada clase CSE a lo largo de la jornada, permitiendo identificar patrones de acumulación horaria. Los contadores individuales por tipo de evento y el total acumulado complementan esta vista agregada.



Fig 42. Gráfica de distribución de eventos y contadores

La tabla de series temporales registra el historial completo de eventos con marca de tiempo, tipo de evento, coordenadas y nivel de confianza, facilitando la trazabilidad y el análisis post-recorrido. El sistema recibió y procesó correctamente la telemetría proveniente del nodo IoT, evidenciando la viabilidad del flujo edge-to-cloud propuesto en la arquitectura de cuatro capas.

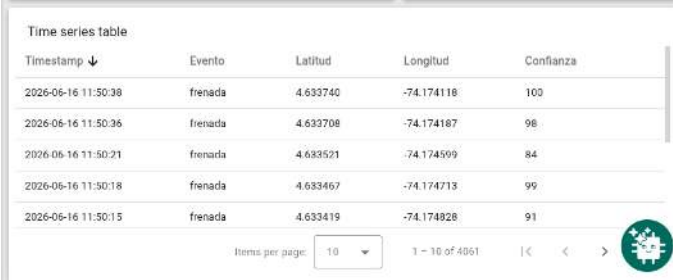


Fig 43. Tabla con el registro histórico de eventos

XIV. RESULTADOS EXPERIMENTALES Y VALIDACIÓN

Con el objetivo de evaluar la precisión, sensibilidad y, en general, la robustez del nodo IoT desarrollado, se realizaron pruebas en un entorno controlado dentro del casco urbano. Este experimento tuvo como finalidad contrastar los eventos físicos reales ejecutados por un ciclista frente a las alertas efectivamente procesadas en el borde por el nodo sensor y registradas en el dashboard de ThingsBoard.

Estos eventos se contaron y validaron manualmente en un recorrido de 15 minutos, los resultados fueron los siguientes

Tipo de evento	Eventos reales	Eventos captados	Número de errores	Tasa de acierto
Bache	8	6	2	75%
Freno de emergencia	10	16	6	62%
Esquivada	10	6	4	60%
Promedio				65.67%

Tabla 5. Comparación entre eventos reales y eventos captados por el nodo IoT

Como se puede observar el sistema IoT logra captar la mayoría de eventos, pero persisten algunos problemas como el exceso en los reportes de freno de emergencia. Estos problemas pueden surgir debido a la diferencia en la naturaleza de los sensores entre el nodo inicial con el smartphone (mediante el cual se entrenó el modelo) y los sensores usados con el ESP32.

XV. CONSIDERACIONES ÉTICAS

La recopilación del dataset de este proyecto estuvo a cargo de uno de los miembros del proyecto, que se ofreció como voluntario. Sin embargo, una clara extensión de este proyecto sería reclutar más voluntarios, así como volverlo disponible mediante una aplicación móvil. Por ser este un sistema de monitoreo, basado en telemetría inercial y posicionamiento global (GPS), implica responsabilidades éticas importantes como garantizar la privacidad de las personas y su integridad física. Por ello, este tipo de extensiones al proyecto deben seguir los siguientes principios:

- **Privacidad y anonimización de datos.**

La recolección de coordenadas GPS puede revelar información sensible, como rutinas diarias, lugares de residencia o sitios de trabajo de los ciclistas. Debido a esto se debe buscar que todos los datos obtenidos mediante Sensor Logger o nodos IoT locales sean anonimizados de forma estricta. Antes de entrenar el modelo se debe eliminar cualquier información de identificación personal (PII), y los trayectos deben ser segmentados para evitar que pueda reconstruirse una ruta completa de origen a destino.

- **Consentimiento informado.**

En la fase de recolección de datos en una ciudad, cada ciclista que quiera participar debe ser informado de manera clara y transparente sobre el propósito del estudio, los sensores utilizados, el tipo de información que se almacenará y el tratamiento que se le dará. La participación debe ser totalmente voluntaria, y cada persona debe poder retirar sus datos del conjunto de entrenamiento en cualquier momento, sin ninguna consecuencia.

- **Seguridad e integridad física durante la recolección.**

Durante la recolección se deben respetar normas de tránsito y criterios de seguridad. Las maniobras artificiales que este proyecto utilizó para generar baches, esquivadas y frenadas de emergencia se realizaron en ambientes controlados, con baja afluencia de personas y vehículos, evitando caídas intencionales o maniobras que comprometieran la integridad física del miembro del equipo que se ofreció como voluntario. Cualquier trabajo similar debe garantizar las mismas condiciones.

- **Mitigación de sesgos geográficos y socioeconómicos.**

Las ciudades, y en particular Bogotá, presentan grandes diferencias en infraestructura ciclista y estado de la malla vial según la localidad. Para evitar que el modelo de aprendizaje automático o los mapas de calor resultantes reflejen únicamente ciertas zonas, la recolección de datos debe buscar cubrir distintas áreas de la ciudad a analizar (norte, centro, occidente y sur). El objetivo es que el sistema no pueda tener sesgos ante ciertos riesgos presentes en sectores periféricos o vulnerables.

- **Transparencia y uso responsable de la inteligencia artificial.**

El modelo entrenado y los mapas generados tienen un propósito estrictamente preventivo e investigativo. El código fuente está disponible como software abierto en el repositorio de GitHub correspondiente, promoviendo la auditoría algorítmica y la colaboración académica. Se descarta de manera explícita cualquier uso del sistema con fines de vigilancia, perfilamiento punitivo de ciclistas o comercialización no autorizada de datos de movilidad.

XVI. ENLACES A CÓDIGO FUENTE

Código Fuente: <https://github.com/w1sec0d/CyclingSafety>

También es posible acceder al notebook de Google Colab mediante el siguiente enlace:
<https://drive.google.com/file/d/1r8x8tvX7PESK64zzi6L-PNHfPNiibNz/view?usp=sharing>

Tener en cuenta el enlace al dataset propio utilizado:
<https://drive.google.com/drive/folders/1-24wbZkLPmLJy7BUMyjm0R-Bd0dZiIx?usp=sharing>. En él se encuentran los datos utilizados (crudos y procesados) dentro de la carpeta *data*.

XVII. CONCLUSIONES Y TRABAJO FUTURO

Hallazgos Principales

- **Factibilidad del uso de sensores disponibles para gran parte de la población:** trabajos anteriores en la literatura utilizaron sensores especiales de GPS con alta frecuencia de muestreo, en contraste con las frecuencias de 1Hz disponibles en celulares comunes. Este trabajo muestra que se pueden obtener resultados competitivos con sensores inerciales (acelerómetro y giroscopio) disponibles en smartphones de gama media como el Xiaomi Poco X6 y con software especializado de bajo costo basados en microcontroladores como el ESP32.
- **Superioridad de la CNN 1D en entornos supervisados:** Los resultados en el conjunto de prueba oculto (evaluado mediante partición por recorridos completos para garantizar independencia estadística) demuestran que, en el paradigma supervisado, la CNN 1D manual supera al MLP. Mientras el MLP alcanzó un F1 macro de 0,8738, la CNN manual logró un 0,9312. Esto valida la hipótesis de que las arquitecturas convolucionales logran capturar las "firmas físicas" o patrones cinemáticos temporales de eventos críticos (como la forma de onda de un bache o una esquivada) que el resumen estadístico del MLP inevitablemente descarta.
- **Diferencia entre desempeño heurístico y supervisado.** En el paradigma heurístico, el MLP obtuvo mejores resultados que la CNN 1D (F1 macro = 0,9627 frente a 0,7953), porque las pseudoetiquetas fueron generadas a partir de scores estadísticos alineados con las features del MLP. En el paradigma supervisado, la CNN 1D superó al MLP (F1 macro = 0,9411 frente a 0,8928), lo que confirma que las arquitecturas profundas capturan patrones temporales que se pierden al resumir la señal en estadísticas agregadas.
- **Robustez metodológica mediante partición por rutas:** La implementación de la partición de datos basada en recorridos enteros, en lugar de ventanas aleatorias, en conjunto con un split correcto del dataset en subconjuntos de entrenamiento, validación y prueba permitió eliminar la fuga de datos (*data leakage*) temporal. Esta decisión metodológica permite asegurar que los resultados presentados (accuracy > 0,98 en CNN supervisada) son una medida real de la capacidad del modelo para generalizar

sobre rutas desconocidas y no una simple memorización de tramos específicos del pavimento.

- **Utilidad del modelo heurístico como baseline y herramienta complementaria.** El modelo heurístico continúa siendo valioso para estimar un `severity_score` continuo y alimentar mapas de calidad de pavimento, incluso cuando no existen anotaciones manuales densas. Sin embargo, para clasificación fina de eventos reales, los resultados finales favorecen el uso de modelos supervisados y, en particular, de la CNN 1D sobre señal cruda.
- **Generación de mapas de riesgo ciclista.** La integración de las coordenadas GPS con las predicciones supervisadas y el `severity score` heurístico permitió producir mapas interactivos de eventos CSE, severidad promedio por celda geográfica y capas combinadas. En los mapas generados sobre 36 recorridos naturales en Bogotá se identificaron zonas con concentración de eventos y tramos de pavimento de peor calidad, cumpliendo directamente con el objetivo de identificación de zonas de riesgo.
- **Implementación funcional de la capa de visualización IoT.** El dashboard configurado en ThingsBoard recibió y procesó correctamente la telemetría MQTT proveniente del nodo ESP32, validando el flujo completo de la arquitectura de cuatro capas propuesta. La integración edge-to-cloud demostró ser operativa bajo condiciones reales de recorrido urbano en Bogotá, con latencia de publicación sub-segundo y correcta georreferenciación de eventos CSE sobre mapas en tiempo real.

Dificultades enfrentadas y soluciones

- **Condiciones cambiantes de conectividad y robustez en el envío de datos:** Durante la construcción del nodo IoT se presentaron dificultades con la conexión a internet, muchas veces intermitente que impedía una conexión estable con la plataforma de Thingsboard. Además, se tuvo algunos problemas con el guardado de datos de forma local en la microSD del nodo IoT, lo cual se cree que es debido a poca robustez del adaptador microSD a SD, junto con vibraciones que provocan desconexiones de la memoria. Ante esto puede ser útil el definir procedimientos ante desconexión a la conectividad móvil y el almacenamiento local.
- **Baja precisión en el nodo IoT final:** La precisión de los eventos en el nodo IoT final se redujo considerablemente frente a los resultados con el smartphone. Esto consideramos que puede causarse por diferencias en la manera en cómo se diferencian los sensores del nodo IoT y los sensores del smartphone. Si se entrenase una red exclusivamente usando el nodo IoT, seguramente se obtendrían resultados mucho mejores.
- **Incompatibilidad con el dataset Bike&Safe.** La discrepancia de escala entre Bike&Safe (mediana de aceleración $\approx 4,55 \text{ m/s}^2$) y Sensor Logger en modo lineal (mediana $\approx 2,4 \text{ m/s}^2$) impidió utilizar el dataset público como complemento de entrenamiento. La causa probable es una compensación parcial de gravedad no documentada en Bike&Safe. La solución adoptada fue utilizar

exclusivamente datos propios, lo que incrementó la carga de recolección de datos pero garantizó la coherencia numérica del pipeline.

- **Errores en las anotaciones manuales.** El retardo humano al pulsar el botón de anotación en Sensor Logger y los toques accidentales consecutivos generaban etiquetas desplazadas o duplicadas. Se implementó un preprocesamiento automático de anotaciones con deduplicación temporal (ventana de 2 s) y corrección de instante basada en detección de picos sensoriales, lo que mejoró la alineación entre la etiqueta y el evento real en la señal IMU.
- **Desbalance severo de clases.** Con una proporción aproximada de 12:1 entre ventanas normales y ventanas de evento, los modelos tendían a favorecer la clase normal. Se abordó mediante data augmentation sobre ventanas de señal, SMOTE para los enfoques supervisados y pesos de clase inversamente proporcionales a la frecuencia. En la CNN supervisada, SMOTE expandió el entrenamiento de 9.591 ventanas originales a 28.480 ventanas balanceadas, mejorando el reconocimiento de clases minoritarias.
- **Desafío de escalabilidad y cobertura urbana:** La transición desde un modelo validado en un conjunto controlado de 48 recorridos hacia una herramienta de monitoreo masivo para la infraestructura ciclista de Bogotá presenta un desafío de escalabilidad significativo. Si bien la arquitectura desarrollada ha demostrado robustez en la clasificación de eventos críticos, la cobertura total de la red de ciclorrutas de la ciudad exige un volumen de datos geográficos y temporales que supera la capacidad de recolección manual. Esta limitación inherente a la fase experimental subraya la necesidad de implementar estrategias de *crowdsourcing* o integración con flotas de bicicletas compartidas, necesarias para densificar el muestreo y transformar el sistema de una prueba de concepto a una plataforma de soporte a la toma de decisiones urbanas a escala metropolitana.

Posible trabajo futuro

Existen múltiples maneras como este trabajo puede ser extendido y complementado. Múltiples objetivos e ideas iniciales no pudieron ser completados por limitaciones de tiempo o por salirse del alcance del proyecto actual, pero se podrían realizar en un trabajo futuro.

- **Estudios con más voluntarios y en más áreas:** el montaje experimental, así como los modelos de machine learning, demostraron ser adecuados para recopilar y clasificar datos de incidentes en ciclismo urbano de forma automática y mediante equipo de relativamente fácil disponibilidad. Los mapas de calor lograron el objetivo de plasmar este conocimiento de la ciudad de forma visual. Sin embargo, el hecho de solo contar con una persona para la recopilación de datos llevó a limitaciones en la cantidad de datos obtenidos, así como la diversidad de áreas de la ciudad cubiertas. Trabajos posteriores podrían reclutar voluntarios y utilizar el marco ético desarrollado para así extender el dataset a más áreas de la ciudad o incluso a otras ciudades.

- **Creación de una aplicación:** actualmente todo el código se encuentra en forma de prototipo dentro de un notebook de Jupyter. Las redes neuronales corren en computadores y el mapa de calor es accesible solo dentro del notebook. Un trabajo futuro podría enfocarse en hacer que las redes neuronales logren correr de forma local en los dispositivos móviles, para obtener datos de una manera masiva.
- **Uso de sensores IoT adicionales:** por salirse del alcance del proyecto, solo se recurrió a acelerómetros, giroscopios y GPS (de 1 Hz en este último caso). Pero también se podrían realizar investigaciones que desarrollen modelos basados en sensores de presión de neumático integrados a la bicicleta.
- **Limitaciones de la infraestructura de servicio y escalabilidad:** si bien la arquitectura propuesta fue integrada exitosamente con la plataforma ThingsBoard, es importante reconocer las restricciones que conlleva el plan gratuito. La retención de datos históricos y el límite de mensajes por unidad de tiempo imponen una restricción física al despliegue masivo del sistema. De esta manera, para una implementación a mayor escala, la arquitectura debe transitar hacia un esquema de despliegue en un servidor propio o similar, que permita gestionar el flujo de múltiples nodos operando a 50 Hz, evitando la pérdida de paquetes críticos durante los picos de actividad.

XVIII. USO DE HERRAMIENTAS DE INTELIGENCIA ARTIFICIAL

Para la preparación de esta entrega se utilizaron los modelos de lenguaje Claude Sonnet, Claude Opus (Anthropic) accedidos a través del IDE Cursor, como asistentes de generación y depuración de código. Gemini Nano Banana y ChatGPT fue usado para la creación de imágenes. Su uso se concentró en las siguientes áreas:

- **Generación de código Python:** los bloques de preprocesamiento de datos, ventaneo, feature engineering, entrenamiento del MLP, preparación de datos para CNN 1D, evaluación comparativa, generación de mapas interactivos y creación de scripts para el ESP32 fueron desarrollados con asistencia de modelos de lenguaje. En todos los casos, el código generado fue revisado, ejecutado y corregido por los autores antes de incorporarse al notebook definitivo.
- **Revisión de redacción:** algunas secciones del documento fueron revisadas con asistencia del modelo para mejorar claridad y coherencia, sin delegar la autoría del contenido técnico ni las decisiones de diseño.

El contenido técnico del documento —incluyendo el diseño del dataset, la elección de arquitecturas, la interpretación de resultados y las conclusiones— refleja el análisis y criterio de los autores del proyecto. Toda salida generada por los modelos fue verificada contra los resultados reales del notebook antes de ser incluida en el informe.

REFERENCIAS

- [1] SECRETARIA DISTRITAL DE MOVILIDAD, "ANUARIO DE SINIESTRALIDAD VIAL DE BOGOTÁ 2024," DATOS.MOVILIDADBOGOTA.GOV.CO. [ONLINE]. AVAILABLE: [HTTPS://DATOS.MOVILIDADBOGOTA.GOV.CO/DOCUMENTS/07c7989a57954f91a2c42d792e7efd19/ABOUT](https://datos.movilidadbogota.gov.co/documents/07c7989a57954f91a2c42d792e7efd19/about)
- [2] S. YAQOUB, S. CAFISO, G. MORABITO, AND G. PAPPALARDO, "DETECTION OF ANOMALIES IN CYCLING BEHAVIOR WITH CONVOLUTIONAL NEURAL NETWORK AND DEEP LEARNING," EUROPEAN TRANSPORT RESEARCH REVIEW, VOL. 15, NO. 1, PP. 9-, MAR. 2023, DOI: 10.1186/s12544-023-00583-4.
- [3] G. B. DA SILVA, "BIKE&SAFE DATASET," MENDELEY DATA. [ONLINE]. AVAILABLE: [HTTPS://DATA.MENDELEY.COM/DATASETS/3j9yh8zni4/3](https://data.mendeley.com/datasets/3j9yh8zni4/3)
- [4] F. J. ORDÓÑEZ AND D. ROGGEN, "DEEP CONVOLUTIONAL AND *LSTM* RECURRENT NEURAL NETWORKS FOR MULTIMODAL WEARABLE ACTIVITY RECOGNITION," *SENSORS*, VOL. 16, NO. 1, 2016.
- [5] J. R. KWAPISZ, G. M. WEISS, AND S. A. MOORE, "ACTIVITY RECOGNITION USING CELL PHONE ACCELEROMETERS," *ACM SIGKDD EXPLORATIONS NEWSLETTER*, VOL. 12, NO. 2, PP. 74–82, 2010.
- [6] M. JAÉN-VARGAS ET AL., "A DEEP LEARNING APPROACH TO RECOGNIZE HUMAN ACTIVITY USING INERTIAL SENSORS," *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 2021.
- [7] G. B. DA SILVA, "DETECTION OF ANOMALIES IN CYCLING BEHAVIOR WITH CONVOLUTIONAL NEURAL NETWORK AND DEEP LEARNING," *EUROPEAN TRANSPORT RESEARCH REVIEW*, VOL. 15, NO. 1, MAR. 2023, DOI: 10.1186/s12544-023-00583-4.
- [8] C. ACAR AND A. M. SHKEL, MEMS VIBRATORY GYROSCOPES: STRUCTURAL APPROACHES TO IMPROVE ROBUSTNESS. NEW YORK, NY, USA: SPRINGER, 2009.
- [9] A. BANKS AND R. GUPTA, "MQTT VERSION 3.1.1," OASIS STANDARD, OCT. 2014. [ONLINE]. AVAILABLE: [HTTPS://DOCS.OASIS-OPEN.ORG/MQTT/MQTT/V3.1.1/OS/MQTT-V3.1.1-OS.HTML](https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html)
- [10] P. WARDEN AND D. SITUNAYAKE, TINYML: MACHINE LEARNING WITH TENSORFLOW LITE ON ARDUINO AND ULTRA-LOW-POWER MICROCONTROLLERS. SEBASTOPOL, CA, USA: O'REILLY MEDIA, 2020.
- [11] OPEN GEOSPATIAL CONSORTIUM, "OGC WEB MAP TILE SERVICE IMPLEMENTATION STANDARD," OGC 07-057r7, 2010. [ONLINE]. AVAILABLE: [HTTPS://WWW.OGC.ORG/STANDARD/WMTS/](https://www.ogc.org/standard/wmts/)
- [12] A. H. ALAVI, P. JIAO, W. G. BUTTLAR, AND N. LAJNEF, "INTERNET OF THINGS-ENABLED SMART CITIES: STATE-OF-THE-ART AND FUTURE TRENDS," *MEASUREMENT*, VOL. 129, PP. 589–606, JUN. 2018, DOI: 10.1016/j.measurement.2018.07.067.
- [13] S. S. REKA AND T. DRAGICEVIC, "FUTURE EFFECTUAL ROLE OF ENERGY DELIVERY: A COMPREHENSIVE REVIEW OF INTERNET OF THINGS AND SMART GRID," *RENEWABLE AND SUSTAINABLE ENERGY REVIEWS*, VOL. 91, PP. 90–108, 2018, DOI: 10.1016/j.rser.2018.03.089.
- [14] K. KAUR, "A SURVEY ON INTERNET OF THINGS - ARCHITECTURE, APPLICATIONS, AND FUTURE TRENDS," IN *PROC. 1ST INT. CONF. SECURE CYBER COMPUTING AND COMMUNICATIONS (ICSCCC)*, 2018, PP. 581–583, DOI: 10.1109/ICSCCC.2018.8703341.