



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO

Tecnológico Nacional de México

**Centro Nacional de Investigación
y Desarrollo Tecnológico**

Tesis de Maestría

**Detección de Comportamiento Sospechoso en Videos
de Situaciones en Carretera**

presentada por
Ing. Mauricio Gutiérrez Montor

como requisito para la obtención del grado de
Maestro en Ciencias de la Computación

Director de tesis
Dr. Nimrod González Franco

Codirector de tesis
Dr. Jorge Alberto Fuentes Pacheco

Cuernavaca, Morelos, México. Septiembre de 2026.



Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Mor., **10/AGOSTO/2026**

OFICIO No. DCC/150/2026

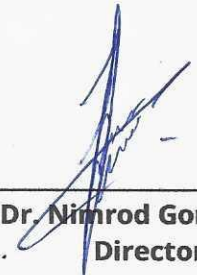
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

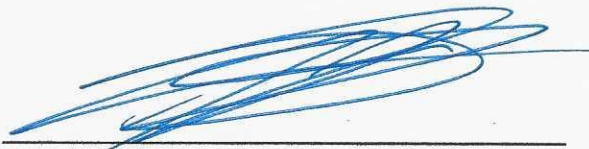
Por este conducto, los integrantes de Comité Tutorial de **Mauricio Gutiérrez Montor** con número de control **M24CE054**, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"Detección de Comportamiento Sospechoso en videos de situaciones en carretera"** y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

ATENTAMENTE

Excelencia en Educación Tecnológica®
"Conocimiento y Tecnología al Servicio de México"


Dr. Nimrod González Franco
Director de tesis


Dr. Jorge Alberto Fuentes Pacheco
Codirector de tesis


Dr. Noé Alejandro Castro Sánchez
Revisor 1


Dr. Juan Gabriel González Serna
Revisor 2

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante



2026
año de
**Margarita
Maza**



Interior Internado Palmira S/N, Col. Palmira,
C.P. 62490, Cuernavaca, Morelos Tel. 01 (777) 3627770, ext. 3201,
e-mail: dcc_cenidet@tecnm.mx | cenidet.tecnm.mx



Centro Nacional de Investigación y Desarrollo Tecnológico
Subdirección Académica

Cuernavaca Mor, 10/agosto/2026

Oficio No. SAC/247/2026

Asunto: Autorización de impresión de tesis

MAURICIO GUTIÉRREZ MONTOR
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS
DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Detección de Comportamiento Sospechoso en videos de situaciones en carretera"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

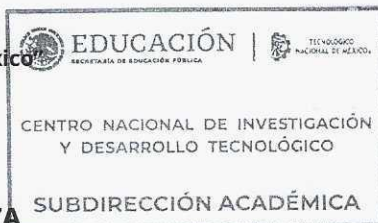
Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

ATENTAMENTE

Excelencia en Educación Tecnológica®

"Conocimiento y Tecnología al servicio de México"

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO



c.c.p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz



2026
año de
Margarita
Maza

cenidet
Centro Nacional de Investigación
y Desarrollo Tecnológico



Dedicatoria

Este trabajo se lo dedico a mi padre Marcelo Gutiérrez Hernández y a mi madre Mirna Oralia Montor Herrera, porque gracias a ellos aprendí a sacar mi mejor esfuerzo por hacer los trabajos con la mayor calidad posible. Gracias por darme los ánimos para seguir adelante. Este es el resultado de su gran labor de enseñanza y siempre se los voy a agradecer.

Este trabajo también está dedicado a los transportistas de México que enfrentan el problema de inseguridad abordado en esta tesis. Espero que la investigación realizada contribuya a sentar un precedente y aporte un granito de arena para mejorar su seguridad en el desempeño de su importante labor.

Agradecimientos

Agradezco a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por el apoyo económico brindado durante mis estudios de maestría. Al Tecnológico Nacional de México / Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET por brindar las instalaciones y permitirme realizar los estudios de maestría. A mis directores de tesis, el Dr. Nimrod González Franco y el Dr. Jorge Alberto Fuentes Pacheco por su guía para llevar a cabo este trabajo.

Resumen

El robo a transporte de carga es un delito de alta incidencia en México que impacta directamente en la seguridad de los transportistas. En el estado del arte se identificó que la mayoría de los desarrollos basados en inteligencia artificial para detectar o reconocer comportamientos sospechosos asociados a delitos se enfocan en cámaras de videovigilancia fijas, ubicadas dentro o fuera de establecimientos, generalmente bajo condiciones controladas. Otros trabajos procesan secuencias en entornos más desafiantes, como carreteras, donde la cámara es móvil y las condiciones de la escena son variables; sin embargo, estos se orientan principalmente a la detección de accidentes. Ante este panorama, en este trabajo se propuso una arquitectura basada en visión artificial para la detección autónoma de comportamientos sospechosos característicos de robo a transporte de carga, utilizando secuencias de video capturadas desde cámaras móviles de tablero (*dashcams*) con vista frontal.

La solución integra módulos de detección de objetos (OD), seguimiento múltiple de objetos (MOT) y clasificación de series de tiempo (TSC) dentro de un *pipeline* asíncrono capaz de operar en tiempo real. Para el entrenamiento y evaluación, se construyeron dos conjuntos de datos novedosos: uno compuesto por videos reales extraídos de redes sociales y otro formado por videos sintéticos generados mediante motores gráficos e inteligencia artificial generativa. La evaluación experimental demostró que la combinación del modelo OD YOLO11s, el sistema MOT Deep OC-SORT y el modelo TSC HydraMultiRocket alcanzó una exactitud balanceada de 83.21 % en la clasificación de eventos normales, sospechosos y evidentes. Con ello, el trabajo realizado constituye un componente funcional para el desarrollo futuro de sistemas de alerta proactiva capaces de anticipar y registrar situaciones de riesgo en carretera.

Palabras clave: robo a transporte de carga, comportamientos sospechosos, visión artificial, cámara de tablero, detección de objetos, seguimiento de múltiples objetos, clasificación de series de tiempo multivariantes.

Abstract

Cargo theft in freight transportation is a highly prevalent crime in Mexico that directly affects the safety of transport operators. A review of the state of the art showed that most artificial intelligence-based approaches for detecting or recognizing suspicious behaviors associated with criminal activity rely on fixed video surveillance cameras, typically located inside or outside facilities under controlled conditions. Other studies process video sequences in more challenging environments, such as highways with mobile cameras and variable scene conditions; however, these approaches are mainly focused on accident detection. To address this gap, this work proposed a computer vision-based architecture for the autonomous detection of suspicious behaviors associated with cargo theft in freight transportation, using video sequences captured from forward-facing dashcams.

The developed solution integrates object detection, multiple object tracking (MOT), and multivariate time series classification models within an asynchronous pipeline capable of real-time operation. For training and evaluation, two novel datasets were created: one comprising real-world videos sourced from social media, and another consisting of synthetic videos generated using graphics engines and generative artificial intelligence. The experimental evaluation showed that the combination of the YOLO11s detector, the Deep OC-SORT multiple object tracking subsystem, and the HydraMultiRocket time series classification model achieved a balanced accuracy of 83.21 % in classifying events as normal, suspicious, or evident. Consequently, the work carried out constitutes a functional component for the future development of proactive alert systems capable of anticipating and recording on-road risk situations.

Keywords: cargo theft in freight transportation, suspicious behaviors, computer vision, dashcam, object detection, multiple object tracking, multivariate time series classification.

Índice general

Resumen	I
Abstract	II
Índice general	III
Índice de Figuras	VIII
Índice de Tablas	IX
Notación, convenciones y terminología	x
1. Introducción	1
1.1. Planteamiento del problema	1
1.2. Delimitación del problema	1
1.3. Complejidad del problema	2
1.4. Objetivos	3
1.5. Alcances y limitaciones	3
1.6. Justificación	4
1.7. Estructura del documento	6
2. Marco Referencial	9
2.1. Fundamentos conceptuales	9
2.1.1. Serie de tiempo multivariante	9
2.1.2. Definición de comportamiento sospechoso	10

Índice general

2.1.3. Seguimiento de múltiples objetos (MOT)	12
2.2. Antecedentes	14
2.2.1. Detección automática de casos de robo a transportistas usando redes neuronales	14
2.2.2. Identificación de armas y personas en carretera bajo condiciones adversas	15
2.3. Estado del Arte	16
2.3.1. Detección de comportamientos sospechosos en cámaras de videovigilancia fijas	17
2.3.2. Detección de accidentes en situaciones de carretera	22
2.3.3. Arquitecturas ML o DL para la clasificación de series de tiempo	24
2.3.4. Arquitecturas MOT	26
2.3.5. Discusión	27
2.4. Estado de la Técnica	28
2.5. Discusión	30
3. Diseño e implementación de la propuesta de solución	31
3.1. Planteamiento de la solución	31
3.2. Heurísticas para la generación de una serie de tiempo a partir de un video	33
3.3. Representación gráfica de una serie de tiempo	35
3.4. Mecanismo de ventana deslizante para homologar la longitud de las series de tiempo	36
3.5. Mecanismo de permutaciones de <i>slots</i> para aumentar el <i>dataset</i> de series de tiempo	40
3.6. Integración de OD, MOT y TSC en <i>pipeline</i> para la inferencia en tiempo real	41
3.7. Implementación por computadora de la propuesta de solución	44
3.7.1. Herramienta para la generación de <i>datasets</i> de series de tiempo	44
3.7.2. Herramienta para el entrenamiento de modelos de clasificación de series de tiempo	46
3.7.3. Herramienta para la prueba del <i>pipeline</i> en tiempo real	48
3.8. Detalles prácticos sobre la implementación	49
4. Experimentación y Resultados	51
4.1. Experimento 1	52
4.1.1. Diseño experimental	52

4.1.2. Ejecución del experimento	53
4.1.3. Resultados y discusión	54
4.2. Experimento 2	55
4.2.1. Diseño experimental	55
4.2.2. Ejecución del experimento	56
4.2.3. Resultados y discusión	56
4.3. Experimento 3	57
4.3.1. Diseño experimental	57
4.3.2. Ejecución del experimento	57
4.3.3. Resultados y discusión	58
5. Conclusiones	61
6. Trabajo futuro	65
Anexo A. Especificación del <i>dataset</i> de videos reales	69
Anexo B. Especificación del <i>dataset</i> de videos sintéticos	72
Referencias	75

Índice de figuras

1.1. Comparativa histórica de la incidencia de robos a transportistas y evolución de la cifra negra.	5
1.2. Actividades que las empresas realizan para operar, independientemente de su giro comercial, que consideran como las más inseguras.	5
1.3. Estado o ubicación del vehículo de carga al momento de efectuarse el robo.	6
2.1. Representación gráfica de una serie de tiempo multivariante.	9
2.2. Situaciones sospechosas que pueden anticipar un robo a transporte de carga.	10
2.3. Representación de los cuatro momentos y tres segmentos de una grabación de un crimen.	12
2.4. Comparación de las arquitecturas SDE y JDE de los sistemas <i>Multiple Object Tracking</i> basados en cinemática y apariencia	14
2.5. Centro de monitoreo Centinela 2.0 desarrollado por ANERPV y WebMaps	29
2.6. Torres de inspección vehicular desarrolladas por la Guardia Nacional	30
3.1. Diagrama arquitectónico de la propuesta de solución.	31
3.2. Esquema conceptual de la integración en <i>pipeline</i> de los módulos OD, MOT y TSC	33
3.3. Ejemplo de la representación gráfica de una serie de tiempo.	36
3.4. Mecanismo de ventana deslizante	39
3.5. Representación temporal del <i>pipeline</i> asíncrono productor-consumidor	43
3.6. Árbol de directorios de los archivos generados por la herramienta para la generación de <i>datasets</i> de series de tiempo.	46
3.7. Árbol de directorios de los archivos generados por la herramienta para el entrenamiento de modelos TSC.	48
3.8. Interfaz gráfica de la herramienta para la prueba del <i>pipeline</i> en tiempo real.	49

Índice de figuras

4.1. Matriz de confusión del <i>fold</i> 4 de 5 en el entrenamiento de HydraMultiRocket en el <i>dataset</i> TSDS1 con el esquema <i>StratifiedGroup 5-Fold Cross Validation</i> . <i>Balanced Accuracy</i> de 0.7695	55
4.2. Matriz de confusión del entrenamiento de HydraMultiRocket en el <i>dataset</i> TSDS3. <i>Balanced Accuracy</i> de 0.8321	59
A.1. Fotogramas representativos de dos videos de cada una de las clases (<i>normal</i> , <i>suspicious</i> , <i>evident</i>) del <i>dataset</i> de videos reales.	69
A.2. Distribución de las clases del <i>dataset</i> de videos reales (versión 4) con 245 videos.	70
B.1. Distribución de las clases del <i>dataset</i> de videos sintéticos (versión 4) con 32 videos.	73
B.2. <i>Prompt</i> utilizado y fotogramas clave del video “Veo E4.mp4” del <i>dataset</i> de videos sintéticos.	74
B.3. <i>Prompt</i> utilizado y fotogramas clave de un video generado en Veo 3 Fast no incluido en el <i>dataset</i> de videos sintéticos.	74

Índice de tablas

2.1. Tabla comparativa del estado del arte del eje temático 1	18
2.2. Tabla comparativa del estado del arte del eje temático 2	22
2.3. Comparación de modelos y arquitecturas.	25
3.1. Bibliotecas principales utilizadas en el desarrollo de las herramientas.	50
4.1. Equipo de cómputo utilizado para la ejecución de los experimentos	53
4.2. Distribución de las clases y subconjuntos del <i>dataset</i> TSDS1 (sin aumento de datos) . .	53
4.3. Resultados del entrenamiento de modelos de clasificación con el <i>dataset</i> TSDS1	54
4.4. Distribución de las clases y subconjuntos del <i>dataset</i> TSDS2 (con aumento de datos por intercambio de <i>slots</i>)	56
4.5. Resultados del entrenamiento de modelos de clasificación con el <i>dataset</i> TSDS2 (<i>StratifiedGroup 5-Fold</i>)	56
4.6. Distribución de las clases y subconjuntos del <i>dataset</i> TSDS3 (<i>Hold-Out</i>)	58
4.7. Resultados del entrenamiento de modelos de clasificación con el <i>dataset</i> TSDS3 (<i>Hold-Out</i>)	58
5.1. Evidencias del cumplimiento de los objetivos específicos	63

Notación, convenciones y terminología

Las ecuaciones se numeran consecutivamente a lo largo del documento. Las figuras y tablas se numeran de manera independiente por capítulo. En las expresiones matemáticas, los delimitadores angulares $\langle z_1, \dots, z_n \rangle$ representan una secuencia finita ordenada, mientras que las llaves $\{z_1, \dots, z_n\}$ representan un conjunto. Por tanto, el orden de los elementos es relevante en el primer caso, pero no en el segundo.

Abreviaturas y siglas

Organizaciones, instituciones y contexto gubernamental

ANERP	Asociación Nacional de Empresas de Rastreo y Protección Vehicular
CANACAR	Cámara Nacional del Autotransporte de Carga
CENIDET	Centro Nacional de Investigación y Desarrollo Tecnológico
ENVE	Encuesta Nacional de Victimización de Empresas (INEGI)
GN	Guardia Nacional
IEEE	<i>Institute of Electrical and Electronics Engineers</i> (Instituto de Ingenieros Eléctricos y Electrónicos)
INEGI	Instituto Nacional de Estadística y Geografía
JCR	<i>Journal Citation Reports</i>
PIB	Producto Interno Bruto
REPUVE	Registro Público Vehicular
RIDFC	Reporte de Incidencia Delictiva del Fuero Común
SESNSP	Secretariado Ejecutivo del Sistema Nacional de Seguridad Pública
UCF	<i>University of Central Florida</i> (Universidad de Florida Central)
UCR	<i>University of California, Riverside</i> (Universidad de California en Riverside)
UEA	<i>University of East Anglia</i> (Universidad de East Anglia)

Seguridad y modelado de eventos

B2NM	<i>Back to Normality Moment</i>
C4	Centro de Comando, Control, Comunicaciones y Cómputo
C5	Centro de Comando, Control, Comunicaciones, Cómputo y Contacto Ciudadano
CCM	<i>Comprehensive Crime Moment</i>
CES	<i>Crime Evidence Segment</i>
FAM	<i>First Appearance Moment</i>
PCBS	<i>Pre-Crime Behavior Segment</i>
SBS	<i>Suspicious Behavior Segment</i>
SCM	<i>Strict Crime Moment</i>

Sistemas de asistencia y monitoreo vehicular

ADAS	<i>Advanced Driver-Assistance Systems</i> (Sistemas Avanzados de Asistencia al Conductor)
CCTV	<i>Closed-Circuit Television</i> (Circuito Cerrado de Televisión)
DMS	<i>Driver Monitoring System</i> (Sistema de Monitoreo del Conductor)
FCW	<i>Forward Collision Warning</i> (Advertencia de Colisión Frontal)
HMW	<i>Headway Monitoring Warning</i> (Advertencia de Monitoreo de Distancia)
LDW	<i>Lane Departure Warning</i> (Advertencia de Salida de Carril)
PCW	<i>Pedestrian Collision Warning</i> (Advertencia de Colisión con Peatones)

Inteligencia artificial, aprendizaje automático y visión artificial

AP	<i>Average Precision</i> (Precisión Promedio)
AUC	<i>Area Under Curve</i> (Área Bajo la Curva)
CNN	<i>Convolutional Neural Network</i> (Red Neuronal Convolutacional)
DL	<i>Deep Learning</i> (Aprendizaje Profundo)
GAN	<i>Generative Adversarial Network</i> (Red Generativa Adversaria)
GRU	<i>Gated Recurrent Unit</i> (Unidad Recurrente con Compuertas)
IA	Inteligencia Artificial
IoU	<i>Intersection over Union</i> (Intersección sobre Unión)
JDE	<i>Joint Detection and Embedding</i>
LPIPS	<i>Learned Perceptual Image Patch Similarity</i>
LSTM	<i>Long Short-Term Memory</i> (Memoria a Largo Corto Plazo)
MAE	<i>Mean Absolute Error</i> (Error Absoluto Medio)
mAP	<i>Mean Average Precision</i> (Precisión Promedio Media)
ML	<i>Machine Learning</i> (Aprendizaje Automático)
MSE	<i>Mean Squared Error</i> (Error Cuadrático Medio)
MOT	<i>Multiple Object Tracking</i> (Seguimiento de Múltiples Objetos)
OD	<i>Object Detection</i> (Detección de Objetos)
PSNR	<i>Peak Signal-to-Noise Ratio</i> (Relación Señal a Ruido Pico)
ReID	Reidentificación
ROI	<i>Region of Interest</i> (Región de Interés)
SDE	<i>Separated Detection and Embedding</i>
SORT	<i>Simple Online and Realtime Tracking</i>
SSD	<i>Single Shot MultiBox Detector</i>
SSIM	<i>Structural Similarity Index</i> (Índice de Similitud Estructural)
VAD	<i>Video Anomaly Detection</i> (Detección de Anomalías en Video)
VAR	<i>Video Anomaly Recognition</i> (Reconocimiento de Anomalías en Video)
YOLO	<i>You Only Look Once</i>

Análisis de series de tiempo

MTS	<i>Multivariate Time Series</i> (Series de Tiempo Multivariadas)
MTSC	<i>Multivariate Time Series Classification</i> (Clasificación de Series de Tiempo Multivariadas)
POTS	<i>Partially Observed Time Series</i> (Serie de Tiempo Parcialmente Observada)
TS	<i>Time Series</i> (Serie de Tiempo)
TSC	<i>Time Series Classification</i> (Clasificación de Series de Tiempo)

Informática, formatos de datos y arquitectura de computadoras

API	<i>Application Programming Interface</i> (Interfaz de Programación de Aplicaciones)
ARFF	<i>Attribute-Relation File Format</i>
CLI	<i>Command Line Interface</i> (Interfaz de Línea de Comandos)
CPU	<i>Central Processing Unit</i> (Unidad Central de Procesamiento)
CSV	<i>Comma Separated Values</i> (Valores Separados por Comas)
GIGO	<i>Garbage-In Garbage-Out</i> (Basura entra, basura sale)
GPU	<i>Graphics Processing Unit</i> (Unidad de Procesamiento Gráfico)
GUI	<i>Graphical User Interface</i> (Interfaz Gráfica de Usuario)
JSON	<i>JavaScript Object Notation</i>
NTSC	<i>National Television System Committee</i> (Formato de video)
PC	<i>Personal Computer</i> (Computadora Personal)
PNG	<i>Portable Network Graphics</i>
RAM	<i>Random Access Memory</i> (Memoria de Acceso Aleatorio)
VRAM	<i>Video Random Access Memory</i> (Memoria de Acceso Aleatorio de Video)

Índice de tablas

Símbolos

x	Coordenada horizontal del centro de un <i>bounding box</i>
y	Coordenada vertical del centro de un <i>bounding box</i>
w	Ancho de un <i>bounding box</i>
h	Alto de un <i>bounding box</i>
A	Cantidad máxima de objetos o <i>slots</i> por cada clase válida de objetos
F	Cantidad de características que describen cada objeto en una observación
D	Cantidad total de características de cada observación de una serie de tiempo multivariante
K	Cantidad de clases válidas de objetos consideradas por el detector de objetos
C	Cantidad de clases de salida del clasificador de series de tiempo
$k_{i,t}$	Índice de la clase asignada por el detector al objeto i en el instante t
$\hat{k}_{i,t}$	Índice de la clase estable del objeto i en el instante t
$H_{i,t}$	Historial ordenado de índices de clase del objeto i hasta el instante t
W	Cantidad máxima de elementos del historial de estabilización de clase
ρ	Proporción mínima requerida para aceptar la clase modal como estable
$\phi_{i,t}(k)$	Frecuencia del índice de clase k en el historial $H_{i,t}$
$\mathcal{M}_{i,t}$	Conjunto de índices de clase que alcanzan la frecuencia máxima en el historial $H_{i,t}$
$k_{i,t}^*$	Índice de la clase modal del historial $H_{i,t}$ cuando esta es única
$\phi_{i,t}^*$	Frecuencia de la clase modal del historial $H_{i,t}$
$\tau_{i,t}$	Frecuencia mínima requerida para actualizar la clase estable del objeto i en el instante t
ℓ	Longitud genérica (cantidad de observaciones) de una serie de tiempo
ℓ_i	Longitud original de la i -ésima serie de tiempo
ℓ_{hom}	Longitud fija a la que se homogeneizan las ventanas de series de tiempo
ℓ_{req}	Longitud fija requerida por la entrada del clasificador de series de tiempo
d_i	Duración temporal, en segundos, del video asociado a la i -ésima serie de tiempo
d_{moda}	Moda de las duraciones temporales de los videos de un conjunto de datos
S	Tamaño de paso (<i>stride</i>) en el mecanismo de ventana deslizante
T_w	Tamaño de la ventana temporal real evaluada (en segundos)
T_i	Intervalo de tiempo entre inferencias sucesivas del <i>pipeline</i> asíncrono
f_{in}	Tasa de fotogramas de entrada; en la homologación fuera de línea se adopta como tasa de referencia común
f_i	Tasa de fotogramas del video asociado a la i -ésima serie de tiempo
f_{eff}	Tasa de fotogramas efectiva tras aplicar el mecanismo de submuestreo
f_{max}	Frecuencia máxima relevante de la dinámica del movimiento
N_{skip}	Cantidad de fotogramas omitidos secuencialmente en la captura asíncrona
t_{in}	Intervalo de tiempo entre la llegada de fotogramas consecutivos del flujo de entrada
t_{gen}	Tiempo promedio de extracción de características (OD + MOT) por fotograma
t_{cls}	Tiempo requerido para una inferencia del clasificador de series de tiempo
TP_k	Cantidad de verdaderos positivos de la clase k
FN_k	Cantidad de falsos negativos de la clase k

Unidades y prefijos

fps	Fotogramas por segundo
px	Píxeles
s	Segundo (Tiempo)
ms	Milisegundo (Tiempo)
b	Bit
B	Byte
bps	Bit por segundo
Bps	Byte por segundo
KB	Kilobyte
MB	Megabyte
GB	Gigabyte

m	Mili (1×10^{-3})
k	Kilo (1×10^3)
M	Mega (1×10^6)
G	Giga (1×10^9)
T	Tera (1×10^{12})

Capítulo 1

Introducción

1.1. Planteamiento del problema

De acuerdo con el Reporte de Incidencia Delictiva del Fuero Común (RIDFC) del Secretariado Ejecutivo del Sistema Nacional de Seguridad Pública (SESNSP), en 2025 se registraron 6263 casos de robo a transporte de carga, de los cuales el 81.76 % implicó violencia (SESNSP, 2026). Por su parte, la Encuesta Nacional de Victimización de Empresas (ENVE) 2024 indica que el 76.1 % de estos delitos no se denuncia ante las fiscalías estatales ni es investigado por estas (INEGI, 2024). Por lo tanto, la cifra real de casos podría ser aún mayor.

De ahí la importancia de desarrollar herramientas computacionales que permitan detectar estos eventos automáticamente. Estas herramientas podrían facilitar la generación de alertas oportunas y fortalecer la evidencia disponible, de modo que las autoridades actúen con mayor rapidez y cuenten con mejores elementos para integrar las carpetas de investigación y proceder contra los responsables.

En la literatura se han desarrollado métodos para detectar o anticipar comportamientos sospechosos a partir de videos captados por cámaras de videovigilancia fijas. Sin embargo, estos métodos suelen estar concebidos para el interior de establecimientos, es decir, para entornos relativamente controlados y con pocos cambios. Por ello, presentan limitaciones para analizar escenas dinámicas en carretera captadas por cámaras móviles.

Por tanto, el problema abordado en esta tesis consiste en **detectar comportamientos sospechosos característicos de un robo a transporte de carga en carreteras.**

1.2. Delimitación del problema

Existen diversos *modus operandi* mediante los cuales los delincuentes ejecutan el robo a transporte de carga, tales como perpetrar el delito en el origen o destino del recorrido, o bien, cuando las unidades se encuentran estacionadas en alguna parte del trayecto. Sin embargo, de acuerdo con Overhaul (2026b), la mayoría de los casos (un 64.1 %) ocurre cuando las unidades están en movimiento, materializándose a través de una intercepción. Por esta razón, abordar este problema requiere un análisis de la dimensión dinámica o temporal; no basta con procesar imágenes estáticas o aisladas, sino que la detección debe realizarse a partir de secuencias de video.

Si bien existen herramientas implementadas en algunas carreteras —como las torres de seguridad que permiten a la Guardia Nacional capturar fotografías de placas para identificar vehículos con reporte de robo o que pudieran ser utilizados para interceptar a los transportistas—, estas presentan la limitación de no abarcar un área territorial suficiente para localizar el punto exacto en el que ocurre un incidente. Aunque esto podría mitigarse parcialmente con el uso de drones (estrategia que ya se aplica en ciertas situaciones), su cobertura sigue siendo restringida. Por lo tanto, resulta fundamental contar con una alternativa basada en el video capturado por cámaras de tablero (*dashcams*), dispositivos que ya se encuentran instalados en múltiples unidades de transporte. La principal ventaja de este enfoque es que los datos generados cuentan con una ubicación geográfica precisa, se obtienen en el momento oportuno y documentan directamente la evidencia visual de cómo suceden los eventos.

Ahora bien, los métodos empleados para interceptar a las unidades en tránsito son variados. Incluyen el establecimiento de retenes falsos, la colocación de obstáculos sobre la vía, la afectación del campo visual del operador (ya sea lanzando piedras al parabrisas o proyectando haces de luz durante la noche) y, de manera muy frecuente, la interposición abrupta de un vehículo en el camino para forzar la detención del transporte.

Por tales motivos, se está delimitando el problema específicamente a trabajar con la detección de comportamientos sospechosos característicos del robo a transporte de carga en carreteras —centrándose en el caso particular de la interposición de un vehículo en el camino—, a partir de videos obtenidos directamente desde cámaras móviles (*dashcams*) instaladas en el vehículo.

1.3. Complejidad del problema

Resolver este problema conlleva múltiples desafíos. En primer lugar, el entorno carretero presenta diversas condiciones adversas que afectan la visión de la cámara, tales como escenas nocturnas, presencia de lluvia, granizo, niebla, nieve o el deslumbramiento frontal del sol. Estos factores reducen los contrastes, introducen ruido en la imagen y atenúan la señal visual debido a los efectos de dispersión y difracción de la luz; en consecuencia, los detalles se difuminan y la visibilidad general disminuye, lo que impacta negativamente en la detección de objetos. En segundo lugar, el uso de secuencias provenientes de *dashcams* implica trabajar con cámaras en constante movimiento, lo que se traduce en una escena altamente dinámica y sujeta a vibraciones, cambios de elevación, entre otras alteraciones. En tercer lugar, las circunstancias en las que ocurre un robo a transporte de carga varían drásticamente de un evento a otro, presentando una cantidad variable de actores involucrados (vehículos, personas, etc.). En conjunto, estos factores complican el reconocimiento de patrones y dificultan el análisis contextual necesario para identificar comportamientos sospechosos en tramos carreteros (Awwad et al., 2024).

Por otra parte, si bien el aprendizaje profundo ha demostrado ser eficaz para analizar imágenes complejas, su rendimiento depende fuertemente de la cantidad y calidad de los datos disponibles para el entrenamiento. En escenarios reales de vigilancia en carretera, obtener suficientes datos visuales para cubrir todas las variaciones posibles de objetos y situaciones resulta una tarea sumamente difícil. La limitada capacidad de los modelos para generalizar a partir de datos insuficientes puede derivar en un rendimiento deficiente ante casos atípicos o no observados durante la fase de entrenamiento, incrementando así la probabilidad de errores en la inferencia (W. Lin et al., 2021).

Adicionalmente, este trabajo requiere extraer y procesar las características espaciotemporales presentes en las secuencias de video, ya que los comportamientos sospechosos —como se mencionó previamente— se manifiestan como patrones dinámicos a lo largo del tiempo. Este análisis demanda la integración de enfoques capaces de capturar la evolución de las interacciones entre los distintos agentes de la escena (vehículos y personas). Sin embargo, procesar esta dimensión temporal aumenta la necesidad

de recursos computacionales y plantea nuevos retos en la optimización del rendimiento del modelo (Chang et al., 2022; T. Wang et al., 2024).

Finalmente, cabe destacar que los modelos de visión artificial basados en aprendizaje profundo son computacionalmente exigentes tanto en la etapa de entrenamiento como en la de inferencia. Durante el entrenamiento —el cual puede prolongarse por días— se requiere procesar un gran volumen de datos para ajustar miles o millones de parámetros en las redes neuronales profundas. Esta fase demanda una considerable cantidad de memoria para almacenar los datos, los mapas de características intermedios y los gradientes calculados durante la retropropagación, un desafío que se agudiza al trabajar con conjuntos de datos extensos y arquitecturas complejas. Asimismo, en la fase de inferencia, la extracción de características y el análisis de las secuencias de video exigen altos recursos computacionales para garantizar una precisión adecuada. Aunque el procesamiento rápido es indispensable para emitir alertas oportunas sobre comportamientos sospechosos en carretera, las limitaciones del *hardware* a bordo del vehículo pueden mermar significativamente la eficiencia del sistema, dificultando su implementación práctica en contextos con restricciones de memoria o capacidad de cómputo.

1.4. Objetivos

El objetivo general de la tesis es:

"Detectar comportamientos sospechosos característicos del robo a transporte de carga a partir de videos mediante técnicas de visión artificial"

Los objetivos específicos se enumeran a continuación.

1. Comparar experimentalmente al menos tres modelos de detección de objetos, dos modelos de seguimiento de objetos y tres modelos de clasificación de series de tiempo, que de acuerdo con el estado del arte resulten más adecuados para resolver el problema.
2. Crear un *dataset* de videos sintéticos, conformado por videos de situaciones sospechosas típicas del robo a transporte de carga y videos de situaciones normales.
3. Crear un *dataset* de videos reales, conformado por videos de situaciones sospechosas típicas del robo a transporte de carga y videos de situaciones normales.
4. Proponer una combinación de técnicas de visión artificial y una arquitectura de aprendizaje profundo para detectar comportamientos sospechosos característicos del robo a transporte de carga.

1.5. Alcances y limitaciones

Los alcances del trabajo de tesis fueron:

1. Implementar modelos ligeros con el fin de que el tiempo de inferencia sea lo menor posible.
2. Desarrollar un prototipo de entrada dual, es decir, que permita la entrada de video en tiempo real desde una cámara física o mediante una cámara virtual.

Capítulo 1. Introducción

3. Generar un *dataset* de videos sintéticos creados mediante simuladores, videojuegos y/o modelos de difusión, en perspectiva primera persona con vista frontal (*front-ego*), organizados en las categorías *evident*, *normal* y *suspicious*.
4. Generar un *dataset* de videos reales provenientes de redes sociales, en perspectiva primera persona con vista frontal (*front-ego*), organizados en las categorías *evident*, *normal* y *suspicious*.
5. Explorar la detección de comportamiento sospechoso en videos que presentan condiciones adversas.

Las limitaciones incluyen:

1. La propuesta se enfocó exclusivamente en videos con vista frontal (*front-ego*).
2. Los videos de los *datasets* no se grabaron manualmente por la inviabilidad y el riesgo.
3. Los videos de los *datasets* se limitaron a carreteras en zonas no urbanas y con baja densidad peatonal.
4. No se implementó un sistema de alertamiento conectado a internet u otra red.
5. El detector se limitó a distinguir entre eventos *evident*, *normal* y *suspicious*, sin identificar qué agentes son los más riesgosos o potenciales participantes del incidente.
6. No se diseñaron nuevos modelos de visión artificial.

1.6. Justificación

El sector de autotransporte de carga desempeña un papel clave en la economía y la sociedad de México. En 2024, contribuyó con el 3.8 % del PIB nacional y movilizó 572 millones de toneladas de productos y mercancías, equivalentes al 58.3 % de la carga doméstica total. Este volumen superó al transportado por los sectores ferroviario, aéreo y marítimo en conjunto (CANACAR, 2025). La flota del sector comprende 723,462 unidades motrices, 711,776 unidades de arrastre y 646 grúas industriales. Estas unidades son operadas por 215,045 permisionarios, de los cuales el 80.26 % son microtransportistas con entre una y cinco unidades motrices y de arrastre (CANACAR, 2025).

Sin embargo, este sector enfrenta importantes problemas de seguridad. Según el RIDFC (SESNSP, 2026), en 2025 se registraron 6,263 robos a transportistas, de los cuales el 81.76 % se cometió con violencia. Estas cifras solo consideran los delitos denunciados ante las fiscalías estatales e investigados por estas. Por su parte, la ENVE 2024 estima que la cifra negra de este delito asciende al 76.1 %, por lo que su incidencia real podría ser mayor (INEGI, 2024). La Figura 1.1 compara los datos más recientes de ambos indicadores con sus antecedentes inmediatos.

El impacto es significativo, especialmente por la violencia implicada, que pone en riesgo la integridad física y la vida de los operadores. Además, como se puede ver en la Figura 1.2, las empresas perciben esta problemática como una de las mayores amenazas: la actividad económica con la que se sienten más inseguras (52.2 % del total de empresas) es el transporte de productos en carreteras, seguida de la movilización de productos en corredores industriales (43.2 %) (INEGI, 2024).

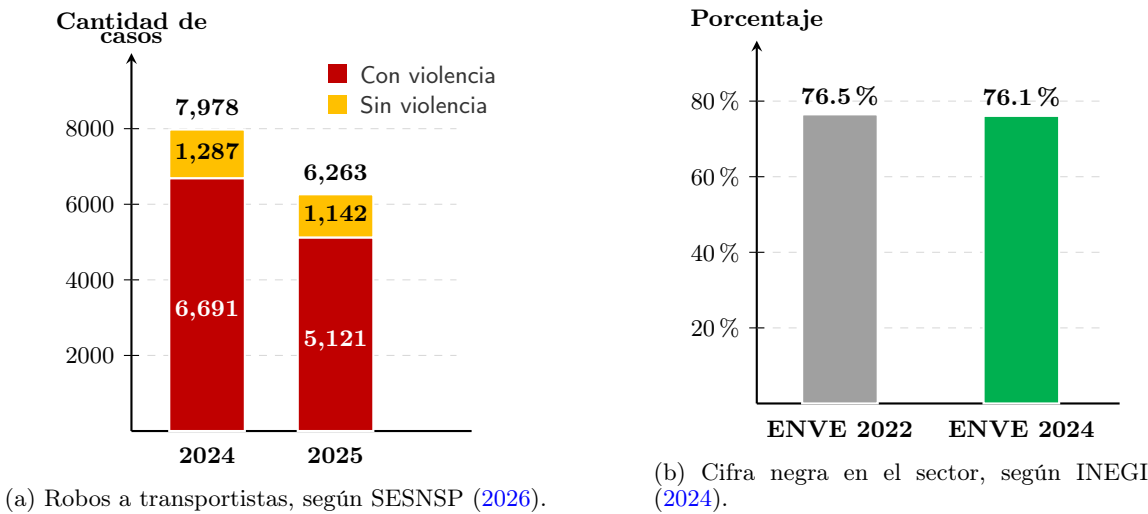


Figura 1.1: Comparativa histórica de la incidencia de robos a transportistas y evolución de la cifra negra.

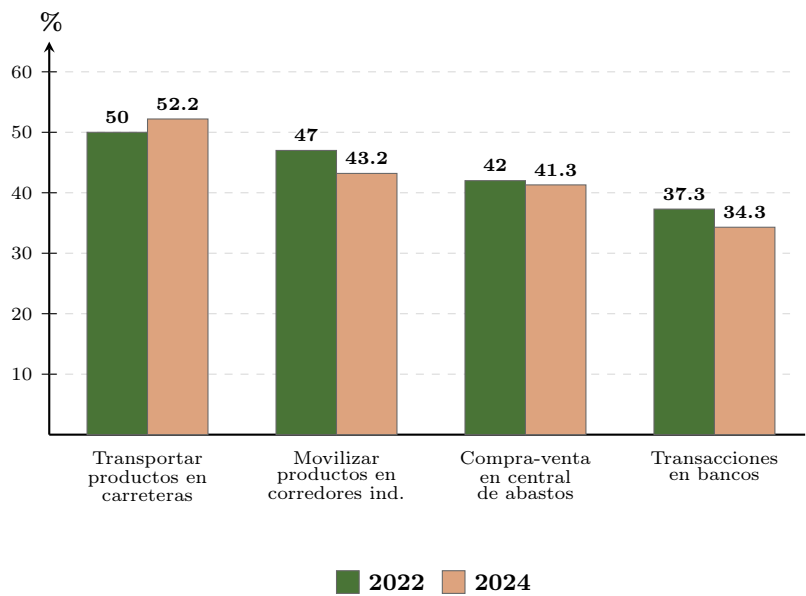


Figura 1.2: Actividades que las empresas realizan para operar, independientemente de su giro comercial, que consideran como las más inseguras. Adaptado de INEGI (2024).

En 2025, el robo a transporte de carga se concentró principalmente en el Estado de México (21 %) y Puebla (17 %), entidades que históricamente han registrado la mayor incidencia de este delito. Los productos más robados fueron alimentos y bebidas (31 %), materiales de construcción e industriales (8 %) y autopartes (8 %). Como se observa en la Figura 1.3, el 64.1 % de los robos ocurrió mientras las unidades estaban en movimiento y el 33.1 % cuando permanecían estacionadas en paraderos para transportistas u otros puntos de detención durante el trayecto (Overhaul, 2026b).

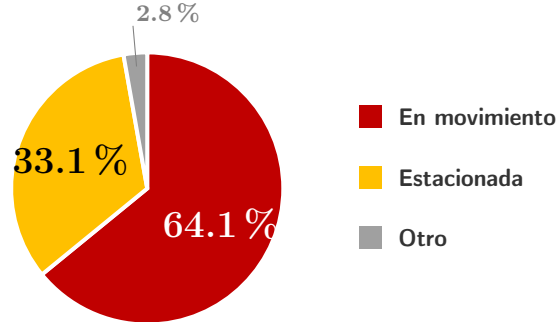


Figura 1.3: Estado o ubicación del vehículo de carga al momento de efectuarse el robo, de acuerdo con Overhaul (2026b).

El problema no solo afecta a las pequeñas empresas. De acuerdo con la ENVE 2024, el robo de mercancías en tránsito es el segundo delito más frecuente sufrido por las grandes empresas, únicamente superado por los robos y asaltos en establecimientos. Además, se encuentra entre los delitos que generan mayores pérdidas económicas, después del secuestro y del robo total de vehículos (INEGI, 2024).

Dado este panorama, la prevención constituye una necesidad urgente. Una estrategia relevante consiste en desarrollar herramientas tecnológicas capaces de detectar oportunamente actividades sospechosas y de apoyar en la emisión de alertas preventivas a las autoridades. En este contexto, un dispositivo basado en visión artificial adquiere especial relevancia. Sin embargo, probablemente no sea suficiente por sí solo: un sistema integral podría requerir información geoespacial y datos procedentes de otros sensores para generar alertas precisas. Por esta razón, la finalidad de este trabajo no es emitir directamente dichas alertas, sino construir un componente de visión artificial que posteriormente pueda incorporarse a un sistema preventivo más amplio.

La detección automática permitiría reducir el tiempo necesario para emitir una alerta. También disminuiría el riesgo de accidentes asociado al alertamiento manual y el peligro de represalias, pues los delincuentes no advertirían que el operador intenta solicitar ayuda. Además, los datos recopilados podrían aprovecharse en el futuro para construir o mejorar un sistema predictivo basado en *big data*.

El desarrollo de esta herramienta resulta especialmente oportuno ante el creciente número de cámaras instaladas en los vehículos. Estas proporcionarían nuevas fuentes de datos que permitirán registrar más evidencias y entrenar modelos de visión artificial con mayor capacidad para detectar comportamientos sospechosos característicos del robo a transporte de carga.

Hasta donde se conoce, se han desarrollado modelos de visión artificial para detectar actividades sospechosas relacionadas con robos en establecimientos. No obstante, no se han identificado modelos dirigidos específicamente a la detección de comportamientos asociados con el robo a transporte de carga en carretera, y menos aún mediante cámaras instaladas en el propio vehículo (Chiuan et al., 2024; Fang et al., 2024). En consecuencia, este trabajo aporta conocimiento relevante en un ámbito todavía poco estudiado.

1.7. Estructura del documento

El presente documento se organiza en seis capítulos. El capítulo 1 introduce y delimita el problema abordado en la tesis, así como los objetivos que guían su desarrollo.

El capítulo 2 presenta el marco referencial, el cual abarca los fundamentos teóricos de los modelos y arquitecturas de *deep learning* aplicados a la detección de objetos (OD), el seguimiento múltiple de objetos (MOT) y la clasificación de series de tiempo (TSC). Asimismo, incluye una revisión del estado del arte y un análisis crítico de las tecnologías actuales de seguridad vehicular en el contexto del transporte de carga.

El capítulo 3 presenta la solución propuesta y el diseño de tres herramientas de *software*: una para transformar videos en series de tiempo multivariantes mediante la detección y el seguimiento de objetos; otra para entrenar modelos de clasificación de dichas series; y una más para integrar los componentes en un *pipeline* de prueba en tiempo real. En este capítulo, los conjuntos de videos reales y sintéticos se presentan únicamente como fuentes de datos y se explica su función dentro del proceso.

El capítulo 4 describe la configuración experimental y presenta los resultados obtenidos al evaluar el desempeño de distintas arquitecturas de detección, seguimiento y clasificación de secuencias temporales.

El capítulo 5 expone las conclusiones de la tesis, junto con la evidencia del cumplimiento de los objetivos planteados.

El capítulo 6 plantea líneas de investigación futuras, orientadas a optimizar la arquitectura y favorecer su viabilidad en entornos operativos.

Posteriormente, los anexos A y B documentan la generación, composición y los criterios de inclusión de los conjuntos de videos reales y sintéticos, respectivamente. Finalmente, la sección de referencias reúne las fuentes bibliográficas citadas en el documento.

Capítulo 2

Marco Referencial

2.1. Fundamentos conceptuales

2.1.1. Serie de tiempo multivariante

Sea una serie de tiempo multivariante X definida como una sucesión de ℓ observaciones en un espacio D -dimensional:

$$X = \langle x_1, x_2, \dots, x_\ell \rangle \in \mathbb{R}^{\ell \times D} \quad (1)$$

donde cada observación individual $x_t \in \mathbb{R}^{1 \times D}$, se compone de $D \geq 2$ características, asociada a una única marca de tiempo s_t (Cao et al., 2018; Du et al., 2023) que indica el instante en que fue obtenida.

Definimos una variable $V_d \in \mathbb{R}^\ell$ como la trayectoria temporal de la d -ésima característica para $d \in \{1, \dots, D\}$.

Como se ilustra en la Figura 2.1, una serie temporal puede representarse mediante la secuencia de variables componentes $\langle V_1, V_2, V_3 \rangle$ y la secuencia de observaciones $\langle x_t, x_{t+\alpha} \rangle$.

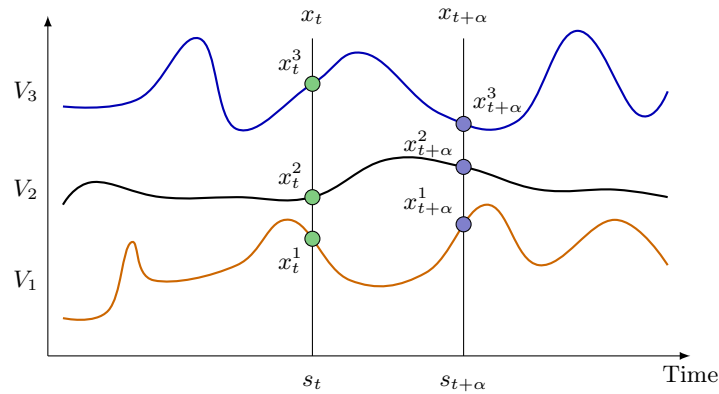


Figura 2.1: Representación gráfica de una serie de tiempo multivariante. Reproducido con permiso de Springer Nature de Gutiérrez Montor et al. (2027).

Formalmente, denotamos como S_d al conjunto de marcas de tiempo registradas para la variable V_d . Bajo

este marco, una serie temporal se clasifica como *regular* si satisface dos condiciones de homogeneidad (IBM Corporation, 2025; X. Zhang et al., 2022):

1. **Sincronía:** Los conjuntos de marcas de tiempo son idénticos para todas las variables:

$$S_i = S_j, \quad \forall i, j \in \{1, \dots, D\} \quad (2)$$

2. **Equidistancia:** El intervalo entre observaciones consecutivas es constante:

$$s_{t+1} - s_t = C, \quad \forall s_t \in S_d \quad (3)$$

Por el contrario, cuando las observaciones carecen del conjunto completo de dimensiones D , la serie se denomina *parcialmente observada* (POTS, por sus siglas en inglés) (Du, 2023). Aunque el procesamiento de estas series puede simplificarse mediante la eliminación de registros incompletos (*listwise deletion*), dicha práctica conlleva una pérdida crítica de información. En su lugar, se emplean métodos de imputación para estimar los valores ausentes basándose en la estructura de correlación interna de la propia serie (Cao et al., 2018).

2.1.2. Definición de comportamiento sospechoso

Las situaciones sospechosas que pueden anticipar el desarrollo de un robo a transporte de carga, y que son visibles desde una cámara instalada en un vehículo, con vista hacia el frente, se ilustran en la Figura 2.2.

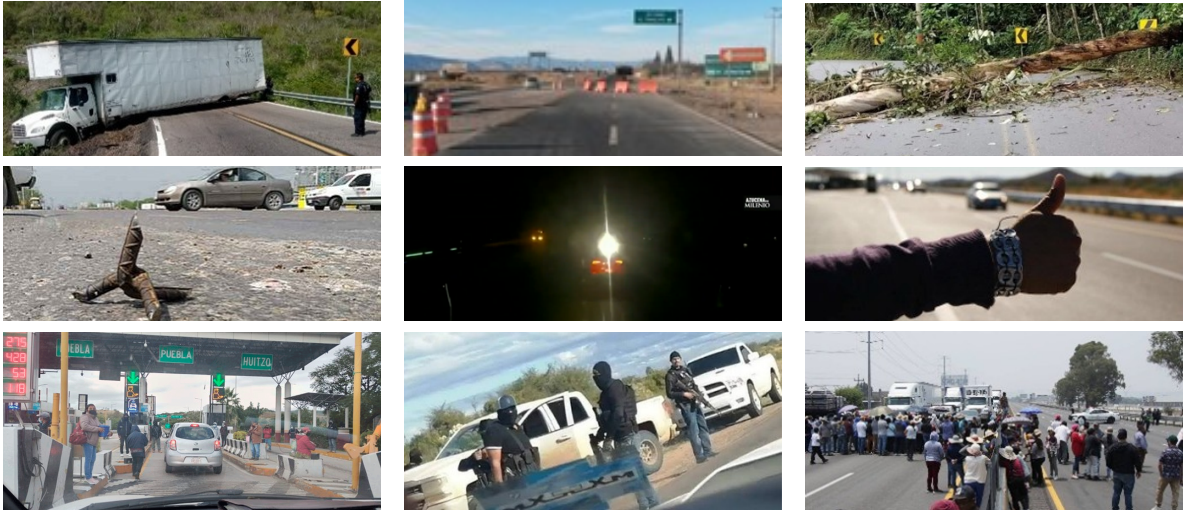


Figura 2.2: Situaciones sospechosas que pueden anticipar un robo a transporte de carga.

Estas situaciones se pueden categorizar de la siguiente forma:

- Bloqueo con objetos estáticos:
 - Objetos convencionales: conos, señalizaciones, barricadas de seguridad, vehículos patrulla atravesados, etc.

- Objetos inusuales: troncos, ponchallantas, entre otros.
- Bloqueo con objetos en movimiento: vehículo interponiéndose en el camino o haciendo frenadas bruscas.
- Afectación del campo visual del operador:
 - Lanzamiento de objetos al parabrisas: piedras, lodo, entre otros.
 - Deslumbramiento: proyección de haz de luz durante la noche o apuntamiento de un láser.
- Escenarios sospechosos con presencia de personas: manifestaciones, retenes falsos, casetas de cobro tomadas por manifestantes, personas solicitando auxilio, raite o atención con intenciones maliciosas.
- Presencia de armas de fuego.

En esta tesis se delimita el análisis a un tipo específico de comportamiento sospechoso: el bloqueo provocado por un vehículo en movimiento. Esta elección se sustenta en que, según Overhaul (2026b), en el 2024 el 76.1 % de los robos a transporte de carga ocurrieron mientras las unidades estaban en movimiento, y la tendencia continuó en 2025 con el 64.1 % de los casos.

Martínez et al. (2021) desarrollaron un trabajo para detectar casos de robo simple (sin violencia) dentro de un establecimiento por medio de visión artificial. En su trabajo dividen el análisis del video en tres segmentos delimitados por cuatro momentos específicos. Los momentos son:

- *First Appearance Moment* (FAM): El individuo que comete el crimen aparece por primera vez en escena.
- *Comprehensive Crime Moment* (CCM): El individuo comienza a actuar de forma sospechosa, por ejemplo, a merodear o voltear de un lado a otro de manera nerviosa.
- *Strict Crime Moment* (SCM): Instante en el que indudablemente el individuo está cometiendo el crimen.
- *Back to Normality Moment* (B2NM): La escena vuelve a un estado visualmente normal.

Los segmentos son:

- *Pre-Crime Behavior Segment* (PCBS): entre FAM y CCM.
- *Suspicious Behavior Segment* (SBS): entre CCM y SCM.
- *Crime Evidence Segment* (CES): entre SCM y B2NM.

Los tres segmentos y cuatro momentos mencionados, se representan en la Figura 2.3.



Figura 2.3: Representación de los cuatro momentos y tres segmentos de una grabación de un crimen. Extraído de Martínez et al. (2021).

Adaptando este esquema al contexto del robo a transporte de carga en carreteras, en esta tesis se definen tres clases de video: *normal*, *suspicious* y *evident*.

- Clase *normal*: grabaciones en las que solo se observan situaciones esperables de una carretera, dentro de los alcances de la tesis.
- Clase *suspicious*: inicia cuando el vehículo implicado entra en escena y muestra un comportamiento anómalo, típicamente al rebasar y frenar de forma brusca con la intención de detener al transportista.
- Clase *evident*: corresponde al momento en que descienden personas del vehículo agresor y se aproximan al transportista para amenazarlo, generalmente portando un arma.

Esta definición difiere de la especificada por Martínez et al. (2021) en dos aspectos centrales:

1. No existe un segmento equivalente al *Pre-Crime Behavior*. En su lugar, la clase *normal* no representa el “antes” de un evento sospechoso, sino grabaciones independientes donde no ocurre ningún incidente.
2. La clase *suspicious* inicia desde la primera aparición del vehículo implicado. En un escenario real captado desde una cámara con vista frontal, el intervalo entre FAM y CCM es prácticamente nulo, por lo que se considera sospechoso desde el instante en que el vehículo ingresa al campo visual con una trayectoria que anticipa la agresión.

2.1.3. Seguimiento de múltiples objetos (MOT)

El seguimiento de múltiples objetos (MOT, por sus siglas en inglés, *Multiple Object Tracking*) es una tarea de visión computacional cuyo objetivo es localizar distintos objetos de interés dentro de una escena y asignarles un identificador único. El reto principal consiste en mantener ese mismo identificador constante a lo largo de los fotogramas consecutivos de un video; es decir, rastrear la trayectoria de cada objeto específico a pesar de que existan oclusiones (cuando un objeto oculta a otro temporalmente), variaciones de escala y movimientos complejos (Erregue et al., 2025).

Para lograr este seguimiento continuo, los sistemas MOT, bajo el paradigma *Tracking-by-Detection*¹, operan mediante dos etapas fundamentales:

- Primero, un modelo de detección localiza los objetos en cada fotograma.
- Posteriormente, un algoritmo de asociación (como el algoritmo húngaro²) vincula estas detecciones individuales para formar trayectorias coherentes a lo largo del tiempo.

Para resolver la etapa de asociación, los modelos pueden basarse exclusivamente en la cinemática de los objetos, es decir, en el cambio de la ubicación y dimensiones de su *bounding box*, o bien, integrar características visuales de apariencia. A continuación, se describe cada una de estas dos opciones:

1. Asociación basada exclusivamente en cinemática

Existen modelos que prescinden totalmente de la extracción de características visuales y confían únicamente en datos espaciales y de movimiento (Y. Zhang et al., 2022).

2. Asociación basada en cinemática y apariencia: incorporación de reidentificación (ReID)

Para resolver asociaciones más complejas y recuperar identidades tras oclusiones prolongadas, los sistemas agregan la técnica de reidentificación (ReID). Esta técnica utiliza redes neuronales profundas para extraer características visuales únicas (*embeddings*) de cada objeto detectado. Dependiendo de cómo se extraigan estos *embeddings*, las arquitecturas se dividen en dos enfoques, los cuales se representan en la Figura 2.4 y se describen a continuación:

- **Arquitectura *Separated Detection and Embedding* (SDE):** En este diseño, la etapa de detección y la de extracción de apariencia son procesos separados. Tras obtener los *bounding boxes*, se realiza una operación de recorte de los objetos detectados, los cuales son procesados posteriormente por una red neuronal independiente (extractor ReID) para obtener sus *embeddings*. Ver Figura 2.4 (a).
- **Arquitectura *Joint Detection and Embedding* (JDE):** Dado que en las arquitecturas SDE, ejecutar un detector seguido de un modelo ReID independiente introduce una alta carga computacional, las arquitecturas JDE buscan optimizar el tiempo de inferencia. Este enfoque unifica la estimación de *bounding boxes* y la extracción de características visuales (*embeddings*) en una única red neuronal. Al compartir los mapas de características de las capas convolucionales para resolver ambas tareas simultáneamente, mediante un *backbone* unificado, los enfoques JDE reducen significativamente el costo de procesamiento. Sin embargo, en la literatura se continúa trabajando para que estos modelos alcancen consistentemente la precisión de asociación que logran las arquitecturas SDE (Erregue et al., 2025). Ver Figura 2.4 (b).

¹Existen otros paradigmas distintos, por ejemplo, *Tracking-by-Segmentation* donde, en lugar de asociar *bounding boxes*, se asocian máscaras de segmentación.

²El algoritmo húngaro fue desarrollado originalmente por Harold Kuhn en 1955 y posteriormente adaptado y formalizado por James Munkres en 1957 (razón por la cual también se le conoce como algoritmo de Kuhn-Munkres). Para ahondar en su desarrollo matemático, se pueden consultar sus respectivas publicaciones: Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2), 83-97; y Munkres, J. (1957). Algorithms for the assignment and transportation types of linear programming problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1), 32-38.

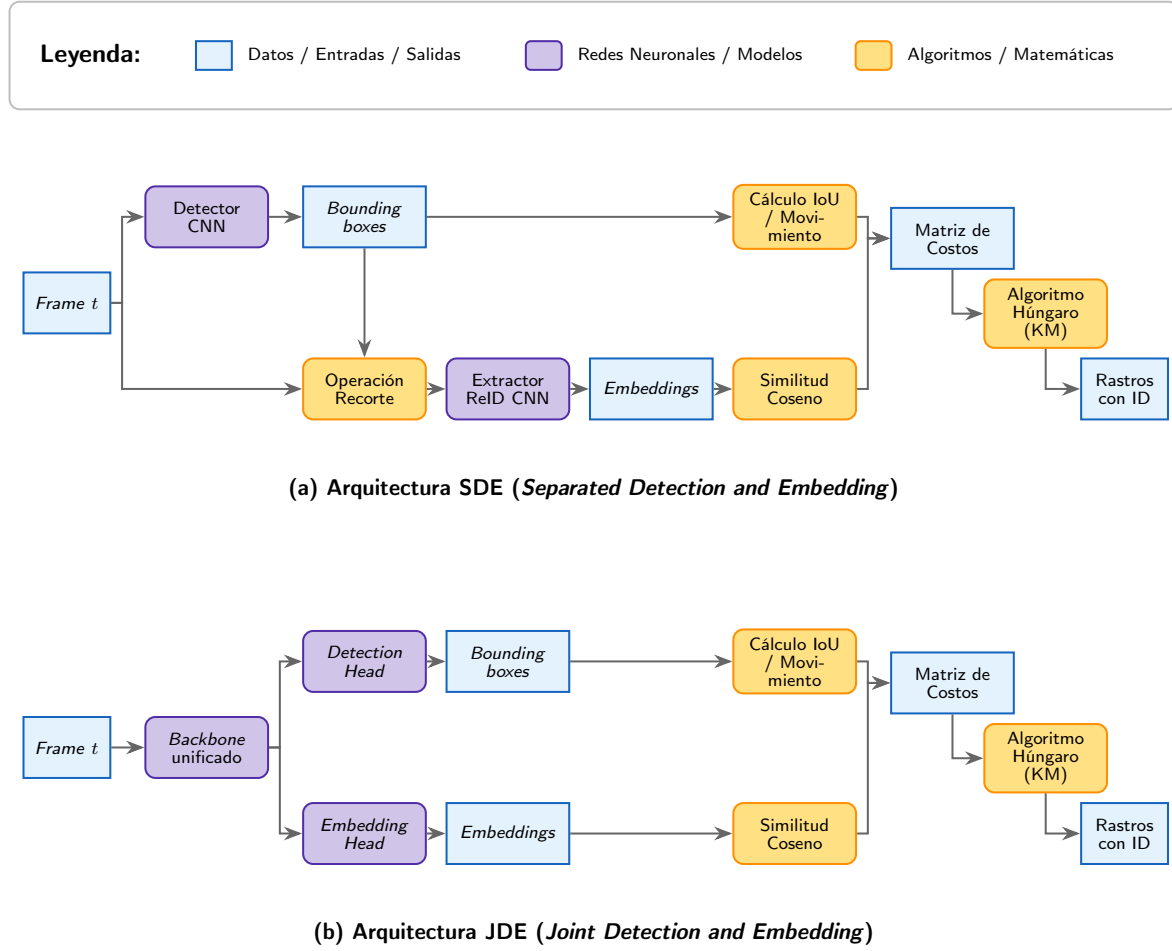


Figura 2.4: Comparación de las arquitecturas SDE y JDE de los sistemas *Multiple Object Tracking* basados en cinemática y apariencia

2.2. Antecedentes

En el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) se han desarrollado dos investigaciones que anteceden directamente a este trabajo. La primera exploró la clasificación de escenas de robo y la detección de armas en videos de carretera. La segunda continuó el estudio de la detección de armas en este entorno y consideró condiciones adversas de iluminación. A continuación, se describen sus principales aportaciones y resultados.

2.2.1. Detección automática de casos de robo a transportistas usando redes neuronales

En el reporte de resultados de su investigación de maestría, Cortés Ramírez (2022) abordó la detección de situaciones sospechosas de robo con violencia a transportistas a partir de videos. Como parte del trabajo, creó el repositorio CRIMEX, integrado por 53,881 imágenes clasificadas como “normal”, “pre-

crimen”, “crimen” y “post-crimen”. Para obtenerlas, desarrolló una herramienta que permitía buscar y descargar videos de YouTube, extraer sus fotogramas, preprocesarlos y etiquetarlos.

Inicialmente, evaluó diferentes arquitecturas de redes neuronales convolucionales para clasificar las imágenes en las categorías crimen y normal. Entre ellas se incluyeron EfficientNet_B0, MobileNet V3, VGG16, VGG19, Xception, ResNet50_V2 e Inception V3. Sin embargo, los resultados de prueba no alcanzaron la precisión esperada. El autor atribuyó esta dificultad a los cambios visuales constantes que presentan las escenas grabadas desde un vehículo en carretera.

Ante estos resultados, la investigación se reorientó hacia la detección de armas mediante SSD y YOLOv5. Para entrenar los modelos se seleccionaron 5,078 imágenes del *dataset* “Detección de Armas – Open Data” del Instituto Andaluz Interuniversitario en *Data Science and Computational Intelligence* (DaSCI). Se excluyeron las imágenes de armas en exhibición y se conservaron aquellas en las que una persona portaba el arma. También se utilizaron 379 imágenes de CRIMEX. Las armas se dividieron en tres clases: arma blanca, arma corta y arma larga.

El modelo SSD, basado en MobileNet V2, se entrenó durante 60,000 pasos. De acuerdo con las curvas presentadas, la pérdida de clasificación se aproxima a una meseta después de aproximadamente 35,000 a 40,000 pasos, con valores cercanos a 0.06. La pérdida de localización muestra un comportamiento similar a partir de aproximadamente 30,000 a 35,000 pasos, con valores entre 0.02 y 0.03. Estas curvas muestran la convergencia del entrenamiento, pero no permiten determinar por sí solas la capacidad de generalización del modelo. Además, la prueba reportada se realizó con una sola imagen y el valor de 94 % corresponde a la confianza de esa detección, por lo que no constituye una evaluación suficiente del rendimiento general de SSD.

La evaluación de YOLOv5 fue más amplia. A partir de las curvas de precisión y *recall*, el modelo obtuvo un AP@0.5 de 0.958 para navaja, 0.848 para arma corta y 0.963 para arma larga. El resultado conjunto fue un mAP@0.5 de 0.923. Estos resultados proporcionan una evaluación más representativa del desempeño del detector en las tres clases consideradas.

Posteriormente, YOLOv5 se integró en la aplicación móvil CriMex 1.0 para realizar pruebas funcionales. Inicialmente se utilizó un umbral de confianza de 0.8, pero este valor ocasionó que se descartaran numerosas detecciones. Por esta razón, el umbral se redujo a 0.3. Esta modificación favoreció el *recall*, aunque también incrementó la presencia de falsos positivos, entre ellos vehículos identificados como armas. Este comportamiento corresponde al compromiso esperado al modificar el umbral de confianza: al reducirlo se admiten más detecciones, mientras que al aumentarlo se conservan únicamente aquellas con mayor confianza.

En algunas partes del reporte, los valores mostrados junto a las *bounding boxes* se relacionan con la precisión del enmarcado y con el IoU. Sin embargo, el código de inferencia utilizado muestra que estas etiquetas contienen la confianza asignada a cada detección. El IoU se emplea por separado durante la supresión no máxima para eliminar cajas redundantes y no se presenta en las etiquetas. Por tanto, los valores observados en estas pruebas corresponden a la confianza del detector y no a una medida del IoU.

2.2.2. Identificación de armas y personas en carretera bajo condiciones adversas

Esta investigación en desarrollo estudia la detección de armas de fuego portadas por personas en videos grabados en carretera mediante YOLO. También considera objetos con una apariencia semejante a la de las armas, con el propósito de reducir los falsos positivos (J. M. Tolentino Morales, comunicación

personal, 2024).

Los videos capturados desde vehículos pueden verse afectados por el movimiento de la cámara, la lluvia y las variaciones de iluminación. De estas condiciones, el trabajo aborda específicamente la baja luminosidad mediante el mejoramiento de imágenes con MIRNet y la evaluación de los resultados con las métricas PSNR, SSIM y LPIPS. Para el entrenamiento y las pruebas se emplean los *datasets* arma-CortaEmpuñada, que contiene imágenes de personas empuñando armas, y DSetNightv3, compuesto por imágenes similares en condiciones nocturnas.

Además, se utiliza Grounding DINO para reducir el trabajo requerido durante el etiquetado de los videos. Finalmente, la investigación propone el diseño de una arquitectura destinada al envío de alertas al C5.

2.3. Estado del Arte

El estado del arte presentado en esta sección se estructura en cuatro ejes temáticos principales:

1. **Detección de comportamientos sospechosos en cámaras de videovigilancia fijas:** Este eje analiza técnicas de visión artificial empleadas para identificar comportamientos anómalos o sospechosos en secuencias de video, principalmente en entornos controlados.
2. **Detección de situaciones sospechosas con cámaras móviles en carretera:** Se investigan enfoques basados en video capturado por cámaras móviles (p. ej., *dashcams*), donde la dinámica del entorno introduce mayor complejidad. La literatura en este contexto se centra principalmente en la detección y anticipación de accidentes, pero también hay trabajos que buscan la detección de obstáculos en la carretera.
3. **Arquitecturas ML o DL para la clasificación de series de tiempo:** A partir de los hallazgos previos, se adopta el modelado de la dinámica temporal del video como series de tiempo multivariantes. Este eje revisa arquitecturas de *machine learning* y *deep learning* para su clasificación.
4. **Arquitecturas MOT:** Se analizan las principales características de los tres sistemas MOT implementados en esta tesis.

Para la búsqueda de artículos se emplearon dos estrategias: revisión sistemática y bola de nieve.

En la revisión sistemática se utilizaron motores de búsqueda y bases de datos como Google Scholar, EBSCO, IEEE Xplore, MDPI y Wiley. Las palabras clave empleadas, organizadas por eje temático, fueron:

- **Primer eje:** *abnormal, anomaly, suspicious, behavior, detection, recognition.*
- **Segundo eje:** *traffic, driving, accident, anomaly, detection, anticipation, dashcam.*
- **Tercer eje:** *time series classification.*
- **Cuarto eje:** *multiple object tracking.*

Mediante la estrategia de bola de nieve, se identificaron trabajos adicionales a partir de *surveys*. En particular, la revisión de la literatura de Chiu et al. (2024) permitió ampliar el primer eje, mientras que el trabajo de Fang et al. (2024) aportó referencias relevantes para el segundo. Para el tercer eje, los artículos se localizaron principalmente a través de recopilaciones e implementaciones disponibles en bibliotecas especializadas como TSAI (Oguiza, 2023), SKTIME (Löning et al., 2019), PyTS (Faouzi & Janati, 2020), PyPOTS (Du, 2023), DL-4-TSC (Ismail Fawaz et al., 2019) y Aeon (Middlehurst et al., 2024).

Para los ejes primero y segundo, tras un filtrado inicial considerando año de publicación (≥ 2020), título, resumen y acceso abierto, se seleccionaron 80 artículos. Posteriormente, se redujo el conjunto a 20 mediante los siguientes criterios:

- **Diversidad de métodos:** se priorizaron enfoques representativos; en caso de propuestas similares, se eligieron las más recientes y con mejoras explícitas sobre trabajos previos.
- **Calidad de publicación:** se dio prioridad a revistas indexadas en JCR.

Como resultado, el conjunto final se compone de:

- 1 artículo de conferencia (IEEE ADESCON) y 19 artículos de investigación publicados a partir de 2021 en revistas JCR, incluyendo *Sensors* (4), *Computation* (1), *IET Image Processing* (2), *Multimedia Tools and Applications* (3), *IEEE Transactions on Intelligent Vehicles* (2), *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2), *Scientific Reports* (1), *Arabian Journal for Science and Engineering* (1), *Neurocomputing* (1), *IEEE Transactions on Parallel and Distributed Systems* (1) e *IEEE Access* (1).
- 2 *surveys* de 2024, uno de ellos en *IEEE Transactions on Circuits and Systems for Video Technology* (JCR).

Para el tercer eje temático, se consideraron exclusivamente arquitecturas de clasificación de series de tiempo, excluyendo trabajos orientados a otras tareas (regresión, imputación, pronóstico o detección de anomalías), y publicados a partir de 2020.

2.3.1. Detección de comportamientos sospechosos en cámaras de videovigilancia fijas

La Tabla 2.1 presenta una comparativa de los trabajos en el estado del arte relacionados con la detección de comportamientos sospechosos en cámaras de videovigilancia fijas (eje temático 1).

Tabla 2.1: Tabla comparativa del estado del arte del eje temático 1

Ref.	Problema	Solución	Dataset	Resultados
(W. Lin et al., 2021)	Contrarrestar el <i>overfitting</i> de modelos de <i>deep learning</i> para detección de anomalías en escenas concurridas debido a la falta de datos.	Para detectar anomalías, se diseñó la red N3D ResNet (basada en <i>3DCNN</i>), entrenada inicialmente con videos sintéticos generados en el videojuego GTA5. Dado que existían sesgos de dominio al evaluar el modelo en el mundo real, se implementó una técnica de adaptación de dominio mediante una <i>3D GAN</i> cíclica (C3DGAN), la cual traduce los datos sintéticos en secuencias fotorealistas. El sistema clasifica los eventos en cuatro niveles de urgencia: <i>normal</i> , <i>need attention</i> (<i>run</i> , <i>chase</i>), <i>need police's help</i> (<i>fight</i> , <i>arrest</i> , <i>knockdown</i>) y <i>public safety</i> (<i>shoot</i> , <i>scatter</i>).	Se creó el <i>dataset</i> sintético SHADE con 2,149 videos, variando clima, horario, fondo y duración (2-34 s). Incluye 9 eventos: <i>arrest</i> , <i>chase</i> , <i>fight</i> , <i>knockdown</i> , <i>run</i> , <i>shoot</i> , <i>scatter</i> , <i>normal1</i> (interacción social) y <i>normal2</i> (caminar). Se armó un <i>dataset</i> de pruebas con 123 videos reales extraídos de UCF-Crime y UMN, con las mismas clases de SHADE salvo <i>normal1</i> .	Se propuso la métrica <i>Misclassification Severity</i> (MISE) para evaluar el error de clasificación penalizando la severidad (urgencia). Un menor MISE indica un mejor rendimiento, evaluado en un rango de [0, 7.5]. En el <i>dataset</i> de videos reales, la exactitud pasó del 17.07 % (antes de la adaptación de dominio) al 26.02 % (después). El valor MISE mejoró de 3.35 a 2.74 tras la adaptación. Aunque los resultados absolutos no son muy favorables, N3D ResNet logra superar a la red base 3D ResNet en el conjunto de pruebas.
(Martínez et al., 2021)	Detectar patrones de comportamiento humano que preceden a las acciones sospechosas evidentes en el robo simple en establecimientos.	Modelo de clasificación binaria (clases <i>normal</i> y <i>suspicious</i>) basado en <i>3DCNN</i> .	Videos de las categorías “ <i>normal</i> ” y “ <i>shoplifting</i> ” del <i>dataset</i> UCF-Crime. La clase “ <i>suspicious</i> ” se construye a partir de <i>clips</i> de “ <i>shoplifting</i> ” recortados usando la metodología propuesta <i>Pre Crime Behavior</i> (PCB).	El modelo con entrenamiento balanceado (120/120) logra 85.71 % y el no balanceado (120/60) 78.26 % de precisión en la clase “ <i>suspicious</i> ”. Se clasifica el video completo, pero no se especifica el momento o persona que ejecuta la actividad sospechosa.
(Mehmood, 2021)	Desarrollar un método de detección de comportamientos sospechosos en escenas poco concurridas (<i>uncrowded</i>) que reduzca la complejidad computacional y requerimientos de memoria con respecto a los métodos hallados en el estado del arte.	En lugar de usar flujo óptico o <i>3DCNN</i> , se obtienen características con el método SG3I y con éstas se entrena una red neuronal <i>2DCNN</i> diseñada (<i>LightAnomalyNet</i>), la cual clasifica entre comportamientos anormales y normales.	■ UR fall ■ Avenue ■ Mini Drone Video ■ Hockey Fights	SG3I alcanzó 757.69 fps . Flujo óptico 16.34 fps . <i>LightAnomalyNet</i> aprende 7154 parámetros, utiliza 14 MB y su latencia es de 23.05 ms en GPU. Inception-V3 utiliza 25 millones de parámetros, 101 MB y su latencia es de 68.6 ms en GPU. En general, consume menos recursos que ResNet-50, Inception-V3 y DenseNet-250; y, aun así, logra 97.42 % de exactitud.
(Ahmed & Echi, 2021)	Desarrollar un sistema que identifique personas que representen una potencial amenaza.	Sistema distribuido de baja latencia. En el <i>edge</i> se encuentran dispositivos como una cámara Logitech C920, Raspberry Pi 3 e Intel NCS2. En este último se ejecuta una CNN de clasificación. En el <i>cloud</i> , se construye una aplicación de usuario y se ejecuta un modelo de segmentación <i>Mask R-CNN</i> .	Se construyó un <i>dataset</i> de imágenes de: granadas, ametralladoras, pistolas, RPG, máscaras, motocicletas, autos, cuchillos y bates. 6,063 de entrenamiento, 2,433 de validación y 1601 de pruebas.	<i>Mask R-CNN</i> logra la mayor exactitud en la clase bate con 96.9 % , y la menor en la clase RPG con 91.4 % . CNN logra la mayor exactitud de 100 % en ametralladoras, y la menor con 90.2 % en autos. El tiempo de predicción promedio de <i>Mask R-CNN</i> es de 4.189 s , y el de CNN de 5.1 ms .

Continúa en la siguiente página...

Tabla 2.1: Tabla comparativa del estado del arte del eje temático 1 (continuación)

Ref.	Problema	Solución	Dataset	Resultados
(Chang et al., 2022)	Detectar comportamientos sospechosos tales como caídas, patadas y puñetazos en el menor tiempo posible a partir de videos.	Sistema multicapa: YOLO v3 detecta personas y extrae características del <i>bounding box</i> (posición, relación de aspecto, altura y sus velocidades de cambio). DeepSORT predice trayectorias evitando <i>ID-switching</i> en oclusiones. Una CNN clasifica acciones por fotograma, y una LSTM detecta comportamientos sospechosos en series de tiempo.	<i>Fall Detection Dataset by ImVia Laboratory of University of Bourgogne</i> (Entrenamiento). <i>UT Interaction Dataset</i> (pruebas).	El modelo logró un desempeño reducido al identificar patadas (sensibilidad 65%, precisión 82%) y puñetazos (sensibilidad 67%, precisión 78%) en el <i>dataset</i> UT Interaction. Con 10 videos de 1 min. (caídas, patadas y puñetazos), se logró una exactitud promedio de 83.31 % y un tiempo de detección promedio de 0.26 s. en una PC con: Intel Core i7-8700K, 32GB DDR4, NVIDIA GeForce 1080Ti.
(Abdullah & Jalal, 2023)	Detectar anomalías en escenarios concurridos mediante el uso de características espaciotemporales y la contabilización y rastreo de personas.	Tras aplicar un filtro de mediana adaptativo (AMF), una segmentación semántica multiclase aísla la clase <i>human</i> en una máscara binaria. Las siluetas se validan como peatones fusionando (promedio ponderado) el Análisis de Movimiento Humano (HMA) y el Modelo de Fuerza de Atracción (AFM). Se ejecuta un conteo con el algoritmo <i>Fuzzy C-Means</i> , y un rastreo combinando el algoritmo húngaro con <i>Dynamic Template Matching</i> . Finalmente, para detectar anomalías (<i>normal/abnormal</i>), un Algoritmo Genético Adaptativo (AGA) optimiza cinco descriptores espaciotemporales que son evaluados por un clasificador neurodifuso multicapa.	<i>Datasets</i> UCSD y Mall para evaluar el conteo y rastreo de personas. <i>Datasets</i> UMN y MED para evaluar la detección de anomalías.	La segmentación semántica logra una exactitud promedio de 96.69 % en UCSD, y 96.03 % en UMN. En el conteo de personas, se logra un MAE ^a de 1.69 y MSE ^b de 2.09 en UCSD; y un MAE de 2.57 y un MSE de 4.34 en Mall. El rastreo de personas, una exactitud prom. de 91.8 % en UCSD y 89.2 % en Mall. En clasificación (<i>normal/abnormal</i>), se logra una exactitud del 93.5 % en UMN, con <i>macro-average</i> de 0.965 en precisión, 0.965 en <i>recall</i> y 0.963 en <i>F1-score</i> . En el <i>dataset</i> MED, se alcanza una exactitud del 92.5 % , con un <i>macro-average</i> de 0.940 en precisión, 0.940 en <i>recall</i> y 0.939 en <i>F1-score</i> .
(Ali, 2023)	Detectar eventos anómalos en videos de videovigilancia en tiempo real, reduciendo la carga computacional y la redundancia espacial al enfocarse únicamente en los objetos en movimiento de la escena.	Sistema AVAD: Emplea Mezcla de Gaussianas (MoG) para sustraer el fondo y aislar el primer plano (objetos en movimiento), que se procesa con un <i>autoencoder</i> convolucional entrenado exclusivamente con eventos normales. Ante comportamientos inusuales, falla, generando un error de reconstrucción que permite clasificar el evento como anómalo y detonar una alarma. En paralelo, YOLOv5 detecta objetos para proporcionar contexto al usuario.	■ UCF-Crime ■ ShanghaiTech ■ CHUK Avenue ■ UCSD-Ped1 ■ UCSD-Ped2	AVAD demuestra gran estabilidad con un AUC promedio de 94.94 % a través de los cinco <i>datasets</i> . Supera a los métodos del estado del arte comparados en UCF-Crime (AUC 97.3 %), CUHK Avenue (96.8 %) y UCSD Ped 1 (91.8 %). En ShanghaiTech logra un 95 % , siendo superado por una red <i>Temporal Branch ConvNet</i> (96.74%). En UCSD Ped 2 obtiene un 93.8 % , siendo superado por un modelo no supervisado basado en GAN (97.2%).

Continúa en la siguiente página...

Tabla 2.1: Tabla comparativa del estado del arte del eje temático 1 (continuación)

Ref.	Problema	Solución	Dataset	Resultados
(Mumtaz et al., 2023)	Reconocer anomalías en videos de videovigilancia a partir de características espaciotemporales, utilizando una técnica de omisión dinámica de fotogramas.	El número de <i>frames</i> a omitir es $F = \frac{V}{N}$, con V la duración del video, y N la duración de la act. sospechosa. Se utiliza un modelo basado en <i>Inflated Inception 3D</i> (I3D) equipado con un bloque de clasificación personalizado (<i>Conv-Top-Layers</i>), el cual sustituye las tradicionales capas de aplanamiento y densas, para reconocer eventos normales y 13 clases de anomalías: <i>abuse, arrest, arson, assault, burglary, explosion, fighting, road accidents, robbery, shooting, shoplifting, stealing y vandalism.</i>	<i>UCF-Crime Dataset</i>	La técnica de omisión de fotogramas ayuda a reducir el desbalance entre las clases. El modelo alcanza un <i>accuracy</i> del 47 % (superando el 40 % que obtenía al no usar las <i>Conv-Top-Layers</i>) y un AUC de 0.837 . El modelo registra un <i>macro-average</i> de 0.38 en precisión, 0.38 en <i>recall</i> y 0.36 en <i>F1-score</i> . Por otro lado, al considerar el peso proporcional de los ejemplos de cada clase (<i>weighted average</i>), el sistema obtiene 0.43 en precisión, 0.47 en <i>recall</i> y 0.44 en <i>F1-score</i> .
(Nazir et al., 2023)	Detectar comportamientos sospechosos antes de que ocurra un robo simple en tiendas a partir de videos de videovigilancia, utilizando características espaciotemporales.	Para la detección y seguimiento de objetos se utiliza YOLOv5 con DeepSORT. Las características del <i>bounding box</i> ingresan a una CNN de clasificación de series de tiempo para identificar comportamientos sospechosos. No se incorpora una CNN para identificar comportamientos en el dominio espacial.	Videos de la categoría “ <i>shoplifting</i> ” y “ <i>normal</i> ” del <i>dataset</i> UCF-Crime.	El mejor modelo de clasificación de series de tiempo basado en CNN fue <i>MiniRocket</i> , con <i>F1-Score</i> medio de 0.89 ± 0.09 , mediano de 0.92 , y precisión de 0.91 ± 0.08 . El esquema propuesto (YOLOv5 + DeepSORT + <i>MiniRocket</i>) sobrepasa en 3 % al <i>F1-Score</i> mediano de RTFM. Además, el tiempo de inferencia total es de 1.6 segundos , mientras que RTFM demora 13.9 s. Las pruebas se ejecutaron en una NVIDIA RTX 2060.
(Xu & Lu, 2023)	Detectar eventos anormales automáticamente (sin necesidad de interpretación humana) en videos de videovigilancia.	Se utiliza una red neuronal convolucional <i>multi-branch</i> (que combina las características y ventajas de Inception y ResNet) para extraer características espaciales que luego son ingresadas por cada fotograma en una GRU para extraer características temporales.	- Exp. Princip.: UCF-Crime (1900 videos de aprox. 240×320px) - Exp. Secund.: UCF-101 (Ambos con prop. <i>train:test</i> de 3:1. Se procesan 50 fotogramas por video)	El modelo logra una exactitud de 86.78 % en UCF-Crime y de 91.17 % en UCF-101. La velocidad de procesamiento de fotogramas es de 157 <i>fps</i> en CPU y 543 <i>fps</i> en GPU. Las pruebas se ejecutaron en una CPU Intel Xeon Gold 5118 y una GPU NVIDIA GeForce RTX 2080Ti.

Continúa en la siguiente página...

Tabla 2.1: Tabla comparativa del estado del arte del eje temático 1 (continuación)

Ref.	Problema	Solución	Dataset	Resultados
(Awwad et al., 2024)	Desarrollar una técnica que mejore imágenes capturadas en condiciones de baja luminosidad para posteriormente detectar anomalías; que pueda ser implementada en dispositivos con recursos limitados.	<i>Pipeline</i> distribuido en nodos <i>edge</i>: ■ Clasificación (Raspberry Pi): Extrae características (filtros de Gabor y Sobel), usa <i>Random Forest</i> para evaluar si la imagen requiere mejora y selecciona la técnica adecuada. ■ Mejoramiento (Dos NVIDIA Jetson Nano): Cada uno implementa una técnica (RUAS o Zero-DCE++), y la ejecutan según la asignación. ■ Detección de objetos (Intel NCS2 - Raspberry Pi): Ejecuta YOLOv5-tiny para identificar los objetos en la imagen final. (Nota: Como <i>baseline</i>, se probó Detectron2 en un entorno <i>Cloud</i> con Intel Xeon Gold 6148 @ 2.10 GHz y NVIDIA Tesla P100 con CUDA 11.5).	Clases <i>people</i> (609 imágenes) y <i>cars</i> (638 imágenes) del <i>dataset</i> ExDark. Cada imagen de cualquiera de las dos clases puede incluir instancias de objetos etiquetados como <i>person</i>, <i>car</i>, <i>bus</i> y <i>motorcycle</i>.	Sin mejoras, Detectron2 obtuvo un <i>mAP</i> de 0.1386 en <i>car</i> y 0.145 en <i>person</i> (clases del <i>dataset</i>, no instancias de objetos). Al aplicar Zero-DCE++ (la mejor técnica), Detectron2 alcanzó un <i>mAP</i> de 0.590 en <i>person</i> (no se reporta dato para <i>car</i>). Con 40 imágenes aleatorias de ambas clases usando el <i>pipeline</i> (ruteo hacia RUAS o Zero-DCE++ y mejoramiento), Detectron2 subió su <i>mAP</i> de 0.4210 a 0.4251 tanto en el <i>edge</i> como en el <i>cloud</i>. Pero su latencia en el <i>edge</i> (4.4 s por imagen frente a 0.09 s en la nube) obligó a utilizar YOLOv5-tiny, cuyo <i>mAP</i> no se reporta. En su implementación final (100% <i>edge</i> con YOLOv5-tiny), las tres fases toman 0.46 s usando RUAS y 0.431 s con Zero-DCE++. Incluyendo la transmisión de red, el ciclo completo exige menos de 1 s por imagen.
(L. Wang et al., 2024)	Reconocer acciones humanas mediante una técnica ligera de detección y predicción de puntos clave de la silueta humana.	Arquitectura basada en tres modelos: ■ LSFE selecciona nodos clave del esqueleto, disminuyendo el tiempo de procesamiento. ■ L-MSFCF predice nodos ocluidos utilizando datos históricos, extrae características a partir de los vectores del esqueleto y sus nodos, y clasifica las acciones humanas. ■ TPT predice trayectorias de nodos representativos (centros de masa de cinco regiones del esqueleto humano: 4 extremidades y región torso-cabeza).	A partir de los <i>datasets</i> Human3.6m, ShanghaiTech Campus y UCF-Crime, se generó un <i>dataset</i> con 2,832 videos de entrenamiento y 314 de pruebas, cada uno con una duración máxima de 10 segundos; organizados en 8 categorías: caminar, correr, detenerse, pelear, patear, escalar paredes, lanzar objetos sospechosos y usar dispositivos cortantes.	LSFE obtiene 76.2% de exactitud (194 ms). L-MSFCF (62.6k parámetros, 162 ms) logra un <i>macro average recall</i> de 0.873. La principal ventaja de incorporar LSFE es la reducción de memoria; sin embargo, el <i>pipeline</i> secuencial resulta más lento (356 ms vs 347 ms) y menos preciso (0.873 vs 0.927) que procesar todos los nodos directamente con la red completa (377.3k parámetros). Finalmente, el modelo TPT (17.6k parámetros, 23 ms) registra tasas de error de 0.19, 0.32 y 0.57 al hacer predicciones de trayectorias sobre 15, 30 y 45 fotogramas, respectiv.
(Taha et al., 2024)	Extraer fotogramas clave en grandes volúmenes de datos de video, para hacer <i>transfer learning</i> a modelos de clasificación pre-entrenados que permitan reconocer eventos anómalos (VAR).	Se desarrolló un método semántico de extracción de fotogramas clave basado en el reconocimiento de acciones y con éstos se hizo un <i>transfer learning</i> a ResNet50, VGG19, EfficientNetB7 y ViT b16 (<i>Vision Transformer</i>) para reconocer eventos anómalos.	<i>UCF-Crime Dataset</i> (entrenamiento y pruebas).	EfficientNetB7 alcanzó un 86.34% de exactitud, VGG19 llegó al 87.90%, ResNet50 obtuvo un 90.46%, y ViT b16 sobresalió con un 95.87% de exactitud.

Continúa en la siguiente página...

Tabla 2.1: Tabla comparativa del estado del arte del eje temático 1 (continuación)

Ref.	Problema	Solución	Dataset	Resultados
(Zeng et al., 2024)	Detectar anomalías conductuales evaluando la divergencia entre el movimiento observado y el predicho a través de tres niveles de granularidad: trayectoria, esqueleto y píxel.	Sistema P ³ VAD que integra tres modelos: una <i>Spatio Temporal Transformer Network</i> (STTN) para predecir trayectorias históricas, una <i>Skeleton Merging Network</i> (SMN) para predecir posturas a nivel esqueleto, y una <i>Skeleton Conditioned Generative Adversarial Network</i> (SCGAN) para predecir regiones de apariencia a nivel píxel utilizando el esqueleto como guía. La métrica de anomalía se obtiene sumando los MSE entre las predicciones y las observaciones a nivel esqueleto y píxel.	- ShanghaiTech Campus - CUHK Avenue - IITB-Corridor - A Day on Campus (ADOC)	Se evalúa: <i>Micro-AUC</i> (calculado sobre la concatenación de todos los fotogramas de todos los videos) y <i>Macro-AUC</i> (promedio de los AUC de cada video). El sistema supera a múltiples métodos del estado del arte, logrando un <i>Micro-AUC</i> / <i>Macro-AUC</i> del 88.74 % / 83.89 % en CUHK Avenue, 84.23 % / 87.09 % en ShanghaiTech, y 74.49 % / 75.53 % en IITB-Corridor. En el <i>dataset</i> ADOC, evaluado únicamente mediante <i>Micro-AUC</i>, alcanza hasta un 92.40 %.

Notas:

^a *Mean Absolute Error* (MAE)

^b *Mean Squared Error* (MSE)

2.3.2. Detección de accidentes en situaciones de carretera

La Tabla 2.2 presenta una comparativa de los trabajos en el estado del arte relacionados con la detección de accidentes en situaciones de carretera mediante cámaras móviles (eje temático 2).

Tabla 2.2: Tabla comparativa del estado del arte del eje temático 2

Ref.	Problema	Solución	Dataset	Resultados
(Yao et al., 2022)	Detectar anomalías de tránsito vehicular en escenas dinámicas a partir de videos de <i>dashcams</i> , utilizando un método no supervisado que evite la necesidad de entrenar con el espectro infinito de posibles accidentes.	Sistema <i>Future Object Localization</i> (FOL), que predice la futura ubicación de los objetos basándose en su trayectoria pasada. Utiliza una arquitectura <i>encoder-decoder</i> y características de <i>bounding boxes</i> (<i>Box</i>), flujo óptico (<i>Flow</i>) y odometría visual (<i>Ego</i>). Una anomalía se detecta cuando la predicción difiere de la realidad, calculando el error mediante tres variantes: FOL-IoU, FOL-Mask y FOL-STD. Se proponen nuevas métricas de evaluación: TARR y STAUC.	Se genera un <i>dataset</i> llamado DoTA con 4677 videos con 18 categorías de anomalías.	El mejor rendimiento lo obtiene <i>FOL-Ensemble</i>, que combina FOL-STD (predicción de objetos) con AnoPred (predicción de fotogramas), alcanzando un AUC de 73.0 % y un STAUC de 48.5 %. Además, gracias a las anotaciones categóricas del <i>dataset</i> DoTA, los autores evalúan métodos complementarios para clasificar el tipo específico de anomalía en videos pregrabados o en tiempo real; sin embargo, su baja precisión demuestra que sigue siendo un desafío abierto.

Continúa en la siguiente página...

Tabla 2.2: Tabla comparativa del estado del arte del eje temático 2 (continuación)

Ref.	Problema	Solución	Dataset	Resultados
(Karim et al., 2024)	Identificar y localizar agentes de tráfico riesgosos en videos de cámaras de tablero (<i>dashcams</i>) antes de potenciales accidentes de tráfico.	Sistema AM-Net, que utiliza: YOLOv5 y DeepSORT para detectar y rastrear agentes de tráfico, generando <i>bounding boxes</i> (<i>Box</i>); RAFT para capturar el flujo óptico a nivel fotograma (<i>Flow-G</i>) y objeto (<i>Flow-O</i>); y ResNet50 para convertir <i>Flow-O</i> y <i>Flow-G</i> en vectores de características. <i>Box</i> (rama 1) y <i>Flow-O/Flow-G</i> (rama 2) se procesan en GRUs independientes para extraer patrones temporales. Un módulo de atención los fusiona, y su salida es decodificada por capas densas y normalizada con <i>softmax</i> , obteniendo una métrica de riesgo: la probabilidad de que un agente <i>i</i> esté involucrado en un accidente en el fotograma <i>t</i> .	Se desarrolló el <i>dataset Risky Object Localization</i> (ROL) con 1000 videos, cada uno de 5 segundos a 20 <i>fps</i> y resolución 1080 x 720. Se hizo un etiquetado espacial, temporal (momento de primera aparición del agente riesgoso y momento del accidente) y categórico.	En el <i>dataset</i> ROL, AM-Net alcanza un AUC de 85.59 % y un tiempo medio para el accidente (mTTA) de 2.18 segundos , demostrando mediante ablaciones que la fusión de características (<i>Box</i> y <i>Flow</i>) junto con la atención dinámica son claves para optimizar este rendimiento. En el <i>dataset</i> DoTA, AM-Net logra el mayor AUC a nivel fotograma (76.5 %), superando a FOL (73.0 %). Finalmente, la eficacia varía según la perspectiva: logra un AUC de 91.23 % (mTTA 2.02 s) en accidentes que involucran al vehículo con cámara (<i>ego-vehicle</i>), y 83.78 % (mTTA 2.79 s) en accidentes de terceros.
(T. Wang et al., 2024)	Predecir accidentes de tráfico de forma precisa y anticipada a partir de videos de <i>dashcams</i> , modelando el estado espacio-temporal de los agentes involucrados (vehículos, peatones, etc.) incluso cuando estos quedan ocultos o salen temporalmente de la escena.	Arquitectura GSC. El módulo <i>Spatio-Temporal Relational Modeling</i> emplea una red convolucional de grafos para modelar las interacciones y trayectorias de los agentes. Si un agente desaparece, el módulo <i>Spatio-Temporal Continuity</i> analiza su historial para determinar si salió de la escena o está oculto; y, en ese caso, predice su ubicación mediante interpolación lineal. El módulo <i>Soft-Attention Aggregation</i> unifica la información de los agentes en una representación global, que un LSTM evalúa para estimar la probabilidad de accidente por fotograma.	■ DAD ■ A3D	Para el <i>dataset</i> DAD, la arquitectura GSC logró una precisión promedio (AP) del 60.4 % , un resultado que superó al estado del arte en su momento. En A3D también se superó a los métodos comparados, con una precisión promedio (AP) de 94.9 % . Sin embargo, en el <i>mean time to accident</i> (mTTA) no superó al mejor método comparado en su estado del arte. En DAD se lograron 2.55 segundos , mientras que en A3D 2.62 segundos .
(Lis et al., 2024)	Detectar obstáculos en la carretera sin necesidad de que el modelo sea entrenado con una infinidad de objetos posibles, partiendo del supuesto de que los objetos fuera del área transitable son irrelevantes para la conducción.	Se delimita el área transitable mediante segmentación semántica y se aplica <i>inpainting</i> mediante ventanas deslizantes para borrar posibles obstáculos, rellenando los píxeles con la textura del entorno. Posteriormente, una red de discrepancia compara la imagen original con la modificada para generar un mapa de calor que resalta las diferencias significativas, indicando la probabilidad de que un píxel pertenezca a un obstáculo.	Para el entrenamiento se utiliza el <i>Cityscapes Dataset</i> , al cual se le superponen objetos sintéticos dentro de la carretera para simular obstáculos. Las pruebas se realizan en los <i>datasets Fishyscapes Lost & Found</i> y <i>RoadObstacles21</i> .	Se evalúa la capacidad para delinear el mapa de calor (AP a nivel píxel) y la eficacia para reconocer el obstáculo como un objeto a esquivar (<i>F1-score</i> a nivel instancia). En el <i>dataset Lost & Found</i> , el método propuesto supera a los algoritmos comparados alcanzando un 82.3 % de AP y un 54.0 % de <i>F1-score</i> . En <i>RoadObstacles21</i> , logra el mejor <i>F1-score</i> (50.8 %), aunque en AP registra un 67.5 % , siendo superado por SynBoost (71.3 %) y Entropy Maximization (85.1 %).

2.3.3. Arquitecturas ML o DL para la clasificación de series de tiempo

Existen varias bibliotecas de código que ofrecen algoritmos y modelos para el procesamiento de series de tiempo (clasificación, imputación, pronóstico o detección de anomalías), entre las cuales destacan TSAI (Oguiza, 2023), SKTIME (Löning et al., 2019), PyTS (Faouzi & Janati, 2020), PyPOTS (Du, 2023), DL-4-TSC (Ismail Fawaz et al., 2019), TSLib (Y. Wang et al., 2025) y Aeon (Middlehurst et al., 2024). Estas siete bibliotecas de modelos están basadas en el lenguaje de programación Python. La Tabla 2.3 (Gutiérrez Montor et al., 2027) presenta una comparación de modelos, arquitecturas y algoritmos específicos de clasificación de series de tiempo publicados a partir de 2020. En ella se definen los siguientes elementos de comparación (Gutiérrez Montor et al., 2027):

- **Exp. (Experimental):** Indica con una marca de verificación (✓) aquellas arquitecturas que fueron seleccionadas explícitamente para ser evaluadas en la fase experimental de este trabajo.
- **Arquitectura:** Nombre de la arquitectura propuesta y su referencia bibliográfica original.
- **Tipo:** Categoría a la que pertenece el modelo según la taxonomía de Faouzi (2022). Para condensar la información, se utilizan las siguientes abreviaturas:
 - **Conv. *Kernels* (Núcleos convolucionales):** Métodos basados en transformaciones utilizando núcleos convolucionales aleatorios. A diferencia del aprendizaje profundo (*deep learning*), y particularmente de las redes neuronales convolucionales, estos métodos no optimizan iterativamente los pesos de sus filtros mediante retropropagación (*backpropagation*); en su lugar, utilizan miles de núcleos estáticos inicializados pseudoaleatoriamente para extraer características estadísticas fijas que posteriormente son procesadas por un clasificador lineal entrenable.
 - **DL: *Deep Learning* (Aprendizaje Profundo).** Arquitecturas basadas en redes neuronales donde los pesos y parámetros de las capas ocultas se actualizan iterativamente durante el entrenamiento mediante el cálculo de gradientes para minimizar una función de pérdida. La arquitectura base subyacente se indica entre paréntesis (p. ej., CNN, *Transformer*, LSTM).
 - **Diccionario:** Enfoques que transforman las series de tiempo en secuencias de símbolos discretos y basan su clasificación en medir la frecuencia de aparición de dichos subpatrones estructurales, construyendo una representación homóloga a una "bolsa de patrones" (*bag-of-patterns*).
 - **Intervalos:** Métodos que extraen características estadísticas temporales y espaciales evaluando subseries de la señal original en diferentes ventanas o intervalos de tiempo, buscando aislar las fases específicas donde las clases son más separables.
 - **Ensamblaje / Híbrido:** Arquitecturas que integran múltiples paradigmas. Los ensamblajes puros combinan las decisiones predictivas de varios clasificadores independientes (votación ponderada), mientras que los métodos híbridos fusionan diferentes algoritmos de extracción de características a nivel de representación antes de someterlos a un único clasificador final.
- **Bibliotecas:** Enumera las bibliotecas de código seleccionadas (TSAI, SKTIME, Aeon, PyTS, PyPOTS) donde el modelo está implementado y disponible para su uso. Los modelos de la biblioteca TSLib (Y. Wang et al., 2025) no se incluyeron debido a su lanzamiento relativamente reciente.
- **Multivar. (Soporte Multivariante):** Establece la capacidad de la arquitectura para procesar series de tiempo con más de una variable ($D \geq 2$).

- **Sí:** El modelo soporta datos multivariantes de forma nativa.
 - **No:** El modelo está diseñado exclusivamente para series univariantes.
 - **Adapt.:** El modelo es originalmente univariante, pero la implementación en la biblioteca citada incluye una adaptación (p. ej., mediante la concatenación de canales o ensamblajes por dimensión) para procesar datos multivariantes.
- **Longitud:** Indica las restricciones sobre la dimensión temporal (ℓ) de las entradas.
- **Fija:** El modelo requiere que todas las instancias en el conjunto de datos tengan exactamente la misma longitud temporal. En este trabajo, la longitud exigida por la entrada del clasificador se denota mediante ℓ_{req} .
 - **Var.:** El modelo tiene la capacidad teórica o técnica para procesar conjuntos de datos que contienen series de tiempo de longitudes temporales variables.

Como se observa en la tabla, la mayoría de las arquitecturas consideradas exige entradas de longitud fija.

- **POTS:** Especifica si la arquitectura puede procesar de forma nativa series de tiempo parcialmente observadas (POTS, por sus siglas en inglés), es decir, datos con valores faltantes o nulos, sin la necesidad de una imputación previa.

Tabla 2.3: Comparación de modelos y arquitecturas. Extraído de Gutiérrez Montor et al., 2027 con permiso de Springer Nature.

Exp.	Arquitectura	Tipo	Bibliotecas	Multivar.	Longitud	POTS
	ROCKET (Dempster et al., 2020)	Conv. <i>Kernels</i>	SKTIME, PyTS, TSAI, aeon	Adapt. ^a	Fija	No
✓	MiniRocket (Dempster et al., 2021)	Conv. <i>Kernels</i>	SKTIME, TSAI, aeon	Adapt. ^a	Fija	No
✓	MultiRocket (Tan et al., 2022)	Conv. <i>Kernels</i>	TSAI, aeon	Adapt. ^a	Fija	No
	POCKET (Chen et al., 2024)	Conv. <i>Kernels</i>	Indep.	No	Fija	No
✓	InceptionTime (Ismail Fawaz et al., 2020)	DL (Inception)	DL-4-TSC, TSAI, aeon	Sí	Fija	No
	LTTE (Ismail-Fawaz et al., 2023)	DL (Inception)	aeon	Adapt. ^a	Fija	No
✓	XCM (Fauvel et al., 2021)	DL (CNN)	TSAI	Sí	Fija	No
✓	Omni-Scale (Tang et al., 2022)	DL (CNN)	TSAI	Sí	Fija	No
	TimesNet (Wu et al., 2023)	DL (Inception)	PyPOTS	Sí	Fija	Adapt.
	TS2Vec (Yue et al., 2022)	DL (CNN)	PyPOTS	Sí	Fija	Sí
	TST (Zerveas et al., 2021)	DL (<i>Transformer</i>)	TSAI	Sí	Fija	Sí
✓	TSSequencer ^c (Tatsunami & Taki, 2022)	DL (LSTM/Seq)	TSAI	Sí	Fija	No
	TSiT (Dosovitskiy et al., 2021)	DL (ViT)	TSAI	Sí	Fija	No
✓	TSPerceiver ^c (Jaegle et al., 2022)	DL (Perceiver)	TSAI	Sí	Fija	No

Continúa en la siguiente página...

Tabla 2.3: Comparación de modelos y arquitecturas (continuación)

Exp.	Arquitectura	Tipo	Bibliotecas	Multivar.	Longitud	POTS
	ConvTran (Foumani et al., 2024)	DL (<i>Transformer</i>)	TSAI	Sí	Fija	No
	CSAI (Qian et al., 2023)	DL (RNN+Trans)	PyPOTS	Sí	Fija	Sí
	iTransformer (Liu et al., 2024)	DL (<i>Transformer</i>)	PyPOTS	Sí	Fija	Adapt.
	HIVE-COTE 2.0 (Middlehurst et al., 2021)	Ensamble	SKTIME, aeon	Sí	Fija	No
✓	Hydra (Dempster et al., 2023)	Conv. <i>Kernels</i> + Diccionario	TSAI, aeon	Adapt. ^a	Fija	No
✓	HydraMultiRocket (Dempster et al., 2023)	Híbrido	TSAI	Adapt. ^a	Fija	No
	WEASEL 2.0 (Schäfer & Leser, 2023)	Diccionario	aeon	No	Fija	No
	Raindrop (X. Zhang et al., 2022)	DL (<i>Graph CNN</i>)	PyPOTS	Sí	Var.	Sí ^b
	Adacket (J. Zhang et al., 2023)	Conv. <i>Kernels</i> (Reinf. <i>Learning</i>)	Indep.	Sí	Fija	No
	TSCF (Dai et al., 2024)	Ensamble	No disp.	No	Fija	No
	TEFN (Zhan et al., 2025)	DL (MLP+ <i>Fuzzy</i>)	PyPOTS	Sí	Fija	Sí
	SAITS (Du et al., 2023)	DL (<i>Attention</i>)	PyPOTS	Sí	Fija	Sí
	QUANT (Dempster et al., 2024)	Intervalos	aeon	Adapt. ^a	Fija	No

Notas:

^a Adaptación disponible para multivariante (originalmente univariante).

^b Soporta un espectro más amplio de series de tiempo irregulares.

^c TSSequencer y TSPerceiver son adaptaciones de Sequencer y Perceiver IO en TSAI.

2.3.4. Arquitecturas MOT

- **ByteTrack (Y. Zhang et al., 2022):** Prescinde de la extracción de características de apariencia (ReID) y basa su funcionamiento exclusivamente en la asociación cinemática (IoU y movimiento). Su aporte principal es el uso de detecciones de baja confianza, las cuales son comúnmente descartadas por otros algoritmos. Mediante un procedimiento secuencial, ByteTrack asocia primero las detecciones con alta probabilidad y, en una segunda pasada, recupera objetos ocluidos o afectados por ruido visual enlazando las trayectorias huérfanas con las detecciones de baja confianza. Esto reduce drásticamente el costo computacional y mantiene una alta precisión de seguimiento.
- **DeepSORT (Wojke et al., 2017):** Extiende el método geométrico SORT al incorporar información de apariencia bajo una arquitectura SDE. Su funcionamiento combina un filtro de Kalman para predecir el estado cinemático, *deep embeddings* para describir la apariencia visual y una métrica de asociación que evalúa la proximidad espacial (distancia de Mahalanobis) y la similitud visual (distancia coseno). La inclusión del ReID le permite mantener trayectorias consistentes en escenas con oclusiones frecuentes.
- **Deep OC-SORT (Maggiolino et al., 2023):** Para mitigar las limitaciones del filtro de Kalman

clásico ante trayectorias no lineales, este modelo se basa en un enfoque centrado en la observación (*observation-centric*). Utiliza las detecciones reales para corregir la acumulación de errores de las predicciones lineales, calculando trayectorias virtuales durante las oclusiones. Deep OC-SORT complementa esta robustez cinemática incorporando un módulo adaptativo de apariencia (ReID) bajo un esquema SDE, mejorando el seguimiento ante aceleraciones bruscas típicas de vehículos.

2.3.5. Discusión

Con base en la revisión del estado del arte, es posible identificar una variedad de formas en la que un comportamiento sospechoso o anomalía es caracterizada. Estas pueden ser:

- **Discrepancia con respecto a predicción:** Primero se hacen varias predicciones sobre un fotograma de un cierto tiempo discreto futuro, con base en los fotogramas presentes conforme avanza el tiempo, y al llegar al tiempo de la predicción se evalúa la discrepancia de las predicciones con respecto al fotograma observado. Dependiendo del nivel de discrepancia, se detectará una anomalía. También se puede detectar una discrepancia a partir de la predicción del movimiento de un objeto (Ali, 2023; Yao et al., 2022).
- **Cambios repentinos en el movimiento:** Mediante flujo óptico se detectan zonas y momentos en el que hay un cambio repentino en el movimiento. Mediante esta caracterización es posible detectar disturbios, escapes, o peleas en zonas concurridas.
- **Series de tiempo anormales:** Se extraen ciertas características en diferentes tiempos discretos, generando una serie de tiempo. Esta serie de tiempo es evaluada para detectar un comportamiento sospechoso (Nazir et al., 2023).
- **Trayectorias inusuales:** Los objetos son rastreados, y las trayectorias resultantes se clasifican con diversos criterios. Algunas serán inusuales, como la persecución y el merodeo, y otras normales (Abdullah & Jalal, 2023; L. Wang et al., 2024).
- **Objetos sospechosos:** Se detecta la presencia de objetos que son inusuales o riesgosos en el contexto de una escena, como armas, herramienta, elementos de camuflaje u obstáculos (Ahmed & Echi, 2021; Lis et al., 2024).
- **Interacción entre objetos:** Se extraen características que relacionan objetos, como la distancia, posición relativa, la cantidad de un grupo de objetos similares, entre otras. En algunas escenas, la presencia de un objeto puede no ser sospechosa, pero la combinación de varios sí lo puede ser (T. Wang et al., 2024).
- **Movimiento del cuerpo humano:** Se detectan comportamientos sospechosos a partir del movimiento del cuerpo humano, para detectar acciones como golpes, caídas, saltos, escalamiento de bardas, entre otros (L. Wang et al., 2024).
- **Escena sospechosa:** se hace una clasificación a nivel de fotogramas, entre sospechoso y no sospechoso (W. Lin et al., 2021; Martínez et al., 2021).
- **Localización del riesgo:** Se busca identificar las zonas de riesgo mediante mecanismos de atención, generándose mapas de fijación. Otro enfoque es localizar a los agentes (objetos) de riesgo en una escena (Karim et al., 2024).

A partir de estas caracterizaciones se identifican tres tipos principales de métodos de aprendizaje: supervisados, no supervisados y semisupervisados. Los métodos supervisados destacan por aprender

tanto comportamientos normales como anormales, e incluso múltiples clases de anomalías; sin embargo, requieren una gran cantidad de datos etiquetados para obtener buenos resultados. Los métodos no supervisados, por otro lado, se centran únicamente en aprender el comportamiento normal y detectan anomalías como desviaciones de este, aunque suelen presentar una mayor tasa de falsos positivos. Ejemplos de ellos son los que evalúan discrepancias con respecto a predicciones. Finalmente, los métodos semisupervisados combinan información etiquetada y no etiquetada, por lo que pueden operar con una cantidad reducida de datos etiquetados y aprovechar un conjunto más amplio de datos sin etiquetar, ofreciendo un equilibrio entre precisión y eficiencia.

Los comportamientos sospechosos suelen ser categorizados de acuerdo con la escena en la que ocurren:

- *Crowded* (concurrido): escenas con una alta densidad de personas u objetos en movimiento dentro del campo de visión de la cámara.
- *Uncrowded* (no concurrido): escenas con baja densidad de personas u objetos en movimiento.

Se observa que la mayoría de los trabajos tienen el objetivo de detectar o reconocer comportamientos sospechosos desde cámaras de videovigilancia fijas, al interior de un establecimiento o fuera de él, ya sea en cruces o avenidas (Chiuan et al., 2024). Si bien se han desarrollado trabajos para detectar o predecir anomalías desde cámaras móviles, únicamente están enfocadas en la detección de accidentes vehiculares (Fang et al., 2024).

La mayoría de los trabajos utilizan el *dataset* UCF-Crime³ para el entrenamiento y pruebas de modelos de detección de comportamientos sospechosos relacionados a delitos. Otros utilizan DoTA⁴ para el entrenamiento y pruebas de anomalías relacionadas a accidentes vehiculares. Sin embargo, estos *datasets* no tienen elementos que coincidan con el fenómeno abordado en la investigación, es decir, no incluyen videos de situaciones sospechosas relacionadas al robo a transporte de carga en carreteras.

2.4. Estado de la Técnica

Existen diversos dispositivos implementados por empresas de transporte de carga para localizar y rastrear sus unidades mediante tecnología satelital. A través de plataformas telemáticas como Samsara o Geotab, estos sistemas permiten, mediante el uso de sensores, monitorear el estado del vehículo, el nivel de combustible y las condiciones de la carga, especialmente cuando se transportan productos que requieren control ambiental, como refrigeración (Geotab, 2026; Samsara, 2026). Entre los dispositivos más utilizados se encuentran las cámaras de tablero (*dashcams*).

La mayoría de las *dashcams* se centran en funciones de grabación continua de la vista frontal y de la cabina. Además, dependiendo del modelo, incorporan sistemas avanzados de asistencia al conductor (ADAS, por sus siglas en inglés, *Advanced Driver-Assistance Systems*) (Navixy, 2026):

- **Forward Collision Warning (FCW):** Advierte sobre una colisión inminente con el vehículo de enfrente a partir del cambio de distancia en el tiempo.
- **Headway Monitoring Warning (HMW):** Monitorea la distancia respecto al vehículo de enfrente y genera alertas cuando se vuelve insegura.

³El *dataset* UCF-Crime puede descargarse desde el sitio web oficial del *Center for Research in Computer Vision* (CRCV) disponible en <https://www.crcv.ucf.edu/projects/real-world/>.

⁴El *dataset* DoTA (*Detection of Traffic Anomaly*) puede descargarse en su repositorio oficial de GitHub disponible en <https://github.com/MoonBlvd/Detection-of-Traffic-Anomaly>.

- *Lane Departure Warning (LDW)*: Detecta salidas involuntarias del carril.
- *Pedestrian Collision Warning (PCW)*: Advierte sobre posibles colisiones con peatones.

Asimismo, integran funciones de monitoreo del conductor (DMS, *Driver Monitoring Systems*), que permiten detectar distracciones, uso de dispositivos móviles, consumo de cigarro o signos de fatiga y somnolencia (Navixy, 2026).

En el ámbito de seguridad, algunas *dashcams* y sistemas de rastreo incorporan botones de pánico físicos. Las *dashcams* suelen integrarse con plataformas web como Numaris, Skyguardian, Webfleet, Samsara y Solera Omnitrac, que permiten a los administradores de flotas visualizar información consolidada, monitorear unidades en tiempo real y gestionar rutas. En este contexto, la Asociación Nacional de Empresas de Rastreo y Protección Vehicular (ANERPV), en conjunto con la empresa WebMaps, desarrolló la plataforma Centinela 2.0 (ver Figura 2.5) (ANERPV, 2026; Indicador Automotriz, 2025).



Figura 2.5: Centro de monitoreo Centinela 2.0 desarrollado por ANERPV y WebMaps. Extraído de Transporte.MX.

Centinela 2.0 conecta aproximadamente 2.5 millones de vehículos en México a través de diversas plataformas de monitoreo (ANERPV, 2026; Indicador Automotriz, 2025). Esta plataforma se integra con centros de comando y control (C4/C5) y con la Guardia Nacional para el seguimiento de incidentes.

De forma complementaria, el gobierno federal, a través de la estrategia BALAM de la Guardia Nacional (ORUSS & CANACAR, 2025; Secretaría de la Defensa Nacional, 2025), ofrece la aplicación móvil *GN Carreteras*, mediante la cual los conductores pueden reportar incidencias o solicitar auxilio desde dispositivos móviles (Guardia Nacional, 2021). La estrategia BALAM contempla el despliegue de recursos operativos en rutas de alta incidencia, incluyendo drones, helicópteros, células de investigación y unidades radiopatrulla conectadas a sistemas de reporte y monitoreo (ORUSS & CANACAR, 2025; Secretaría de la Defensa Nacional, 2025).

Por otro lado, existen empresas que desarrollan análisis estadístico y geoespacial sobre la incidencia delictiva en el transporte de carga a partir de reportes operativos, como Overhaul o AI27 (Gómez, 2023; Overhaul, 2026a). En este sentido, Overhaul ha propuesto el desarrollo de modelos de *deep learning* enfocados en la detección de armas en entornos carreteros, con el objetivo de integrarse en *dashcams* y plataformas de monitoreo como Centinela 2.0 (Indicador Automotriz, 2025; Overhaul, 2026a).

Finalmente, el Área de Innovación y Desarrollo Tecnológico de la Dirección General Científica de la

Guardia Nacional ha desarrollado torres de inspección vehicular equipadas con cámaras capaces de leer placas de hasta 60 vehículos por minuto a velocidades superiores a los 150 km/h (Garlem, 2025) (ver Figura 2.6). Estas torres integran antenas de radiofrecuencia para identificar calcomanías del REPUVE, sistemas de consulta a bases de datos, detectores de *jammers* y conexión satelital. Al 13 de octubre de 2025, se reporta la operación de 23 torres en entidades como Sinaloa, Querétaro y Estado de México, con proyecciones de expansión (Alvarado, 2025).



Figura 2.6: Torres de inspección vehicular desarrolladas por la Guardia Nacional. Extraído de Editora La Prensa (oem.com.mx/la-prensa).

2.5. Discusión

A partir del estado de la técnica, se identifica una brecha importante en la respuesta tecnológica frente al robo a transporte de carga en carreteras. Las *dashcams* actuales (incluyendo sus funciones ADAS y DMS) están orientadas a la seguridad vial y al monitoreo del conductor, pero no contemplan la detección de amenazas delictivas provenientes del entorno exterior. De forma similar, los botones de pánico y la aplicación GN Carreteras dependen de la intervención manual del conductor, lo que limita su uso en escenarios de alto estrés, como un asalto en curso. Las torres de inspección y el monitoreo en centros de comando, aunque amplían la cobertura a nivel infraestructura, mantienen un enfoque principalmente reactivo y sin una perspectiva directa desde el lugar de los hechos.

En este contexto, el sistema autónomo basado en visión artificial propuesto en este trabajo permite detectar el evento de forma automática, lo que abre la posibilidad de generar alertas sin depender de la intervención humana en futuras implementaciones. Aunque actualmente presenta un margen de detección de segundos (lo que limita su capacidad para prevenir el asalto como tal), esta automatización reduce la probabilidad de que un evento no sea reportado por pánico o coacción y permite activar de inmediato acciones clave (como el inicio de grabaciones en alta resolución para mejorar la calidad de la evidencia y la eficiencia en el almacenamiento al guardar únicamente los momentos relevantes).

Además, la captura estructurada de la cinemática del delito introduce una fuente de información que no está presente en los sistemas actuales. Este tipo de datos, generados directamente desde la perspectiva del vehículo y en el momento del evento, puede contribuir a fortalecer el análisis posterior de los incidentes y, a mediano y largo plazo, al desarrollo de modelos predictivos.

Capítulo 3

Diseño e implementación de la propuesta de solución

3.1. Planteamiento de la solución

En la Figura 3.1 se presenta la arquitectura de la solución propuesta.

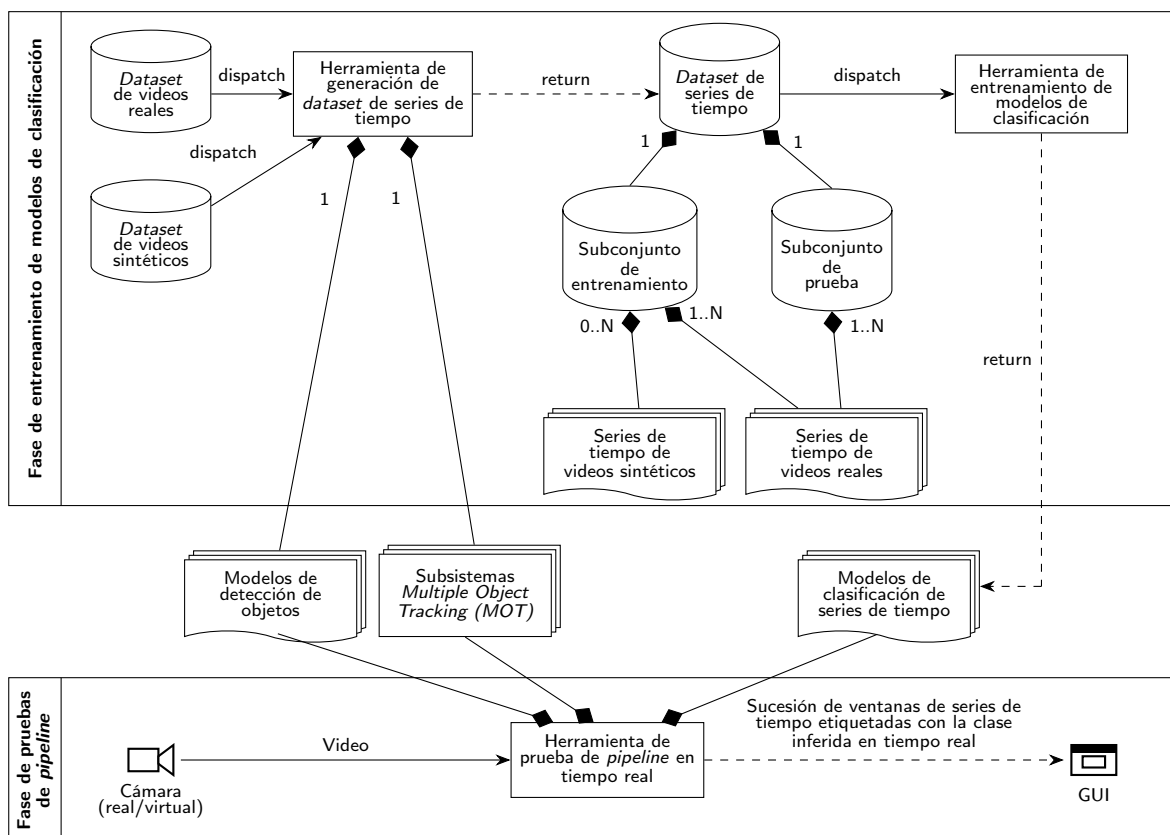


Figura 3.1: Diagrama arquitectónico de la propuesta de solución.

Capítulo 3. Diseño e implementación de la propuesta de solución

Durante la fase de entrenamiento de los modelos de clasificación, se construyeron dos conjuntos de datos:

- **Conjunto de datos de videos reales:** Un *dataset* conformado por videos obtenidos a partir de redes sociales. Su especificación se puede consultar en el Anexo A.
- **Conjunto de datos de videos sintéticos:** Un *dataset* que se generó a partir de la grabación de *gameplays* de los videojuegos GTA V⁵ y American Truck Simulator⁶, así como de la generación de videos usando el modelo Veo 3 Fast⁷. Su especificación se puede consultar en el Anexo B.

Ambos *datasets* incluyen instancias de las clases *normal*, *suspicious* y *evident*, conforme a las definiciones establecidas en la Sección 2.1.2.

A partir de estos conjuntos, la herramienta de generación de *datasets* de series de tiempo —integrada por un módulo de detección de objetos (OD) y otro de seguimiento múltiple de objetos (MOT)— permite extraer series de tiempo para cada video. Con estas series se construyen *datasets* etiquetados, los cuales son homologados en longitud mediante un esquema de ventana deslizante o, posteriormente, aumentados mediante un esquema de permutación de ranuras (*slots*).

En los *datasets* resultantes se respeta rigurosamente la siguiente restricción: el subconjunto de entrenamiento contiene siempre series de tiempo provenientes de videos reales y, opcionalmente, de videos sintéticos; en contraste, el subconjunto de prueba se conforma exclusivamente con series de tiempo derivadas de videos reales. Cada *dataset* generado se utiliza, según el experimento, para entrenar modelos de clasificación de series de tiempo (TSC). La herramienta de entrenamiento de modelos TSC permite implementar esquemas de validación cruzada estratificada y agrupada (*StratifiedGroupKFold*), garantizando que no exista contaminación de datos (*data leakage*) entre conjuntos. Esto asegura que ninguna serie derivada de un mismo video de origen aparezca en más de una partición y que, en todo momento, se cumpla la restricción de excluir datos sintéticos del conjunto de prueba.

En la fase de pruebas, los módulos OD, MOT y TSC se integran en una herramienta de prueba de *pipeline* en tiempo real. Con ella se procesa el video proveniente de una cámara (real o virtual) y se genera como salida en una interfaz gráfica (GUI) una secuencia de ventanas de series de tiempo, las cuales son etiquetadas con la clase inferida en tiempo real.

Cabe señalar que, aunque las herramientas operan utilizando un único modelo o subsistema por categoría en cada ejecución, se implementaron y evaluaron múltiples alternativas. Para OD se emplearon YOLO11s, entrenado con el *dataset* COCO⁸, y YOLO11m, entrenado con el *dataset* BDD100K⁹. Para MOT se implementaron DeepSORT, ByteTrack y Deep OC-SORT. Finalmente, para TSC se utilizaron nueve arquitecturas: HydraMultiRocket (Tan et al., 2022), Hydra (Dempster et al., 2023), Inception-Time (Ismail Fawaz et al., 2020), MiniRocket (Dempster et al., 2021), MultiRocket (Tan et al., 2022), OmniScaleCNN (Tang et al., 2022), TSPerceiver (Jaegle et al., 2022), TSSequencer (Tatsunami & Taki, 2022) y XCM (Fauvel et al., 2021).

⁵La licencia de uso del contenido generado con este videojuego puede consultarse en la página de Rockstar Games: <https://www.rockstargames.com/mx/legal>.

⁶La licencia de uso y distribución de contenido puede consultarse en la página de SCS Software: https://www.scssoft.com/video_sharing_licence.pdf.

⁷Las condiciones del servicio y políticas de uso pueden consultarse en la página de Google: <https://policies.google.com/terms>.

⁸COCO (*Common Objects in Context*) es un conjunto de datos para el reconocimiento de objetos en escenas cotidianas (T.-Y. Lin et al., 2014). Su documentación y descarga se encuentran en <https://cocodataset.org/>.

⁹BDD100K (*Berkeley DeepDrive 100K*) es un conjunto de datos de conducción desarrollado por Berkeley DeepDrive (Yu et al., 2020). Su documentación y herramientas se encuentran en <https://github.com/bdd100k/bdd100k>; la distribución de los archivos se realiza mediante el portal oficial de Berkeley DeepDrive.

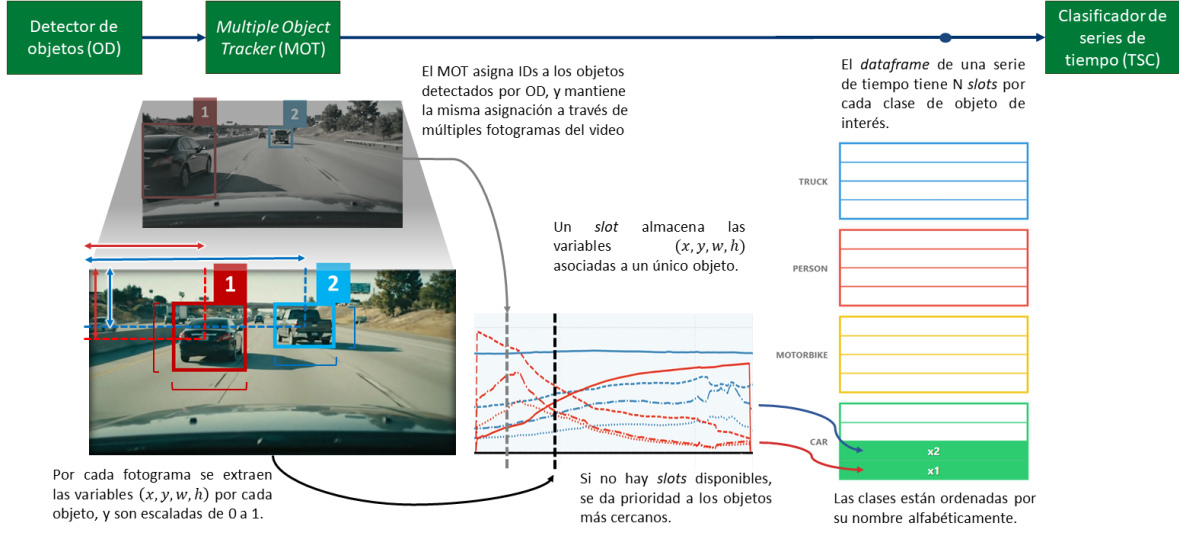


Figura 3.2: Esquema conceptual de la integración en *pipeline* de los módulos OD, MOT y TSC

La Figura 3.2 ilustra la integración de los módulos OD, MOT y TSC en la herramienta de prueba en tiempo real, los cuales operan de la siguiente manera:

1. El módulo OD identifica y localiza los elementos de interés en cada fotograma del video, particularmente automóviles, camiones, motocicletas y personas.
2. El módulo MOT asigna identificadores persistentes a los objetos detectados y, mediante heurísticas, construye series de tiempo multivariantes que describen sus trayectorias.
3. El módulo TSC procesa dichas series para generar inferencias periódicas sobre el comportamiento observado, produciendo como salida las clases *normal*, *suspicious* o *evident*.

Para garantizar el correcto funcionamiento del sistema, fue necesario considerar diversos aspectos técnicos y diseñar un conjunto de heurísticas específicas, las cuales se describen en las siguientes secciones de este capítulo como parte integral de la propuesta de solución.

3.2. Heurísticas para la generación de una serie de tiempo a partir de un video

En el sistema, el archivo de video se separa en fotogramas y se pasa uno a la vez por los módulos OD y MOT. Del *bounding box* de cada objeto detectado y rastreado se extraen la posición x , posición y , ancho w y alto h . Cada una de ellas constituye una variable independiente de la serie de tiempo multivariante. A la secuencia ordenada $\langle x, y, w, h \rangle$ almacenada en el *dataframe* se le denomina *slot*. Cada *slot* contiene las variables de un objeto y, cuando no existe una asignación activa en un fotograma determinado, sus valores permanecen en cero para conservar una dimensionalidad fija. Todas las variables de un objeto son normalizadas en un rango de 0 a 1.

Los objetos detectados y rastreados son únicamente aquellos de las clases válidas que se definan, por ejemplo, *car*, *motorcycle*, *person* y *truck* para el caso de YOLO11s entrenado en COCO y *car*, *truck*,

motor, *person* y *rider* en YOLO11m entrenado en BDD100K. Las clases disponibles y sus nombres difieren entre ambos detectores porque se conservan las taxonomías de sus respectivos conjuntos de entrenamiento. Para cada clase válida se reservan A slots, donde A es la cantidad máxima de objetos de esa clase que pueden detectarse y rastrearse simultáneamente. En el *dataframe*, los slots de cada clase forman un bloque contiguo, y estos bloques se ordenan lexicográficamente según el nombre de la clase.

En una serie de tiempo con un conjunto de $K = 4$ clases válidas y $A = 4$ objetos por cada clase, se tiene un total de $4 \times 4 = 16$ slots. Cada slot tiene $F = 4$ variables correspondientes a la secuencia de características $\langle x, y, w, h \rangle$. Por tanto, la serie de tiempo multivariante tiene una dimensionalidad $D = KAF = (4 \times 4) \times 4 = 64$: 16 slots con 4 variables cada uno. Esta convención permite que todos los fotogramas generen vectores con la misma estructura, independientemente del número real de objetos presentes en la escena.

Entre las heurísticas consideradas en este módulo se encuentran:

- **Filtro de confianza y área mínima:** Antes de enviar las detecciones al módulo MOT, se descartan aquellas que no superan un umbral mínimo de confianza o cuyo *bounding box* ocupa un porcentaje demasiado pequeño de la imagen total. Esto permite eliminar detecciones débiles, ruido visual u objetos muy lejanos que todavía no son relevantes para el análisis, evitando además saturar el sistema con elementos de baja utilidad.
- **Limpieza geométrica de detecciones:** Después de los filtros básicos, se aplica una depuración adicional para reducir detecciones duplicadas o contenidas dentro de otras. Esta etapa incluye una supresión de no máximos agnóstica a la clase y un filtro de contención proporcional, útil para descartar falsas detecciones generadas por regiones internas de un mismo objeto, como puertas, ventanas o fragmentos de vehículos que podrían ser interpretados como objetos independientes.
- **Mecanismo de expulsión de *trackings* (prioridad por tamaño):** Dado que el sistema tiene un número de slots limitado para guardar las series de tiempo, este mecanismo asegura que siempre se siga a los objetos más críticos. Si los slots están llenos y aparece un objeto nuevo, el sistema compara los tamaños: expulsa del rastreo al objeto más pequeño (el más lejano) y le cede su slot al objeto más grande, dándole prioridad a los elementos más cercanos a la cámara.
- **Filtro de cofre (ego-vehículo):** Evita que el sistema rastree por error el cofre del vehículo desde donde se graba. Para lograrlo, evalúa la forma geométrica de la detección (su relación de aspecto) dependiendo de qué tan abajo se encuentre en la imagen. De esta manera, el filtro deja pasar formas cuadradas (que corresponden a otros vehículos frenando muy cerca), pero bloquea las formas muy anchas y horizontales, que son la firma visual típica del cofre. El problema de los falsos positivos en los cofres de los ego-vehículos aparece principalmente en el modelo YOLO11s entrenado en COCO.
- **Mecanismo de estabilización de clase:** Consiste en mantener, para cada identificador de seguimiento, un historial de las clases asignadas por el detector. Su objetivo es reducir los cambios transitorios de etiqueta entre clases visualmente similares. Por ejemplo, una *pick-up* puede detectarse en algunos fotogramas como *truck* y en otros como *car*. Debido a que el *dataframe* reserva para cada clase un bloque compuesto por A slots, una fluctuación momentánea de la etiqueta puede hacer que los datos que representan una misma trayectoria se registren en un bloque de clase distinto. Al mantener una clase estable para cada identificador, el mecanismo evita que su representación se fragmente entre estos bloques. Este procedimiento no modifica la trayectoria estimada por el MOT, sino el bloque del *dataframe* en el que se registran los datos que la representan. El mecanismo se describe en el Algoritmo 1:

Algoritmo 1 Actualización de la clase estable para el objeto i en el tiempo t

Require: Índice de clase detectada $k_{i,t} \in \{1, \dots, K\}$, historial previo $H_{i,t-1}$, índice de clase estable previo $\hat{k}_{i,t-1}$ (si existe), ventana máxima W y proporción mínima de estabilidad ρ

Ensure: Índice de clase estable $\hat{k}_{i,t}$

```

1: Actualizar historial:
2:  $H_{i,t} \leftarrow H_{i,t-1}$ 
3: if  $|H_{i,t}| = W$  then
4:   Eliminar de  $H_{i,t}$  el elemento más antiguo
5: end if
6: Anexar  $k_{i,t}$  al final de  $H_{i,t}$ 
7: Calcular frecuencias:
8: for  $k \in \{1, \dots, K\}$  do
9:    $\phi_{i,t}(k) \leftarrow |\{q \in H_{i,t} \mid q = k\}|$ 
10: end for
11: Determinar las clases modales y su frecuencia:
12:  $\mathcal{M}_{i,t} \leftarrow \arg \max_{k \in \{1, \dots, K\}} \phi_{i,t}(k)$ 
13:  $\phi_{i,t}^* \leftarrow \max_{k \in \{1, \dots, K\}} \phi_{i,t}(k)$ 
14: Calcular umbral dinámico:
15:  $\tau_{i,t} \leftarrow \lceil \rho \cdot |H_{i,t}| \rceil$ 
16: Actualizar clase estable:
17: if  $|H_{i,t-1}| = 0$  then
18:    $\hat{k}_{i,t} \leftarrow k_{i,t}$  ▷ Inicialización
19: else
20:   if  $|\mathcal{M}_{i,t}| = 1 \wedge \phi_{i,t}^* \geq \tau_{i,t}$  then
21:     Sea  $k_{i,t}^*$  el único elemento de  $\mathcal{M}_{i,t}$ 
22:      $\hat{k}_{i,t} \leftarrow k_{i,t}^*$ 
23:   else
24:      $\hat{k}_{i,t} \leftarrow \hat{k}_{i,t-1}$  ▷ Realimentación o empate
25:   end if
26: end if
27: Salida:  $\hat{k}_{i,t}$ 

```

3.3. Representación gráfica de una serie de tiempo

Sea K la cantidad de clases válidas para los objetos detectados, definidas en la serie de tiempo¹⁰. Las series de tiempo resultantes tienen valores acotados entre 0 y 1. Para representar visualmente las variables de las series de tiempo de manera que se facilite su análisis, conviene separarlas en distintas bandas por clase de objeto. Para ello, el rango de 0 a 1 es dividido en K bandas, asignadas a cada una de las clases válidas. Cada banda tiene un espacio de $\frac{1}{K} \times 0.95$ y comienza con un *offset* de $\frac{i}{K}$ con $i \in [0, K-1] \cap \mathbb{Z}$. Las bandas i son asignadas siguiendo un orden lexicográfico a partir del nombre de la clase.

Lo anterior significa que cada variable es escalada en un factor de $\frac{1}{K} \times 0.95$ y a cada *slot* (secuencia ordenada de variables) se le suma el *offset* correspondiente a la clase del objeto al que está representando, para que sus valores queden dentro de la banda asignada.

En las gráficas generadas, los *slots* se diferencian uno de otro dentro de cada banda por el color asignado. Como se puede observar en la Figura 3.3, uno de los *slots* en la banda de la clase *truck* está representado en color naranja. Este *slot* contiene la secuencia de cuatro variables $\langle x, y, w, h \rangle$, las cuales se pueden identificar gracias al tipo de línea asignada y a las acotaciones que aparecen en la esquina inferior derecha.

¹⁰No confundir la cantidad de clases válidas para los objetos detectados por el detector de objetos K , con la cantidad C de clases en las que se clasifican las series de tiempo generadas.

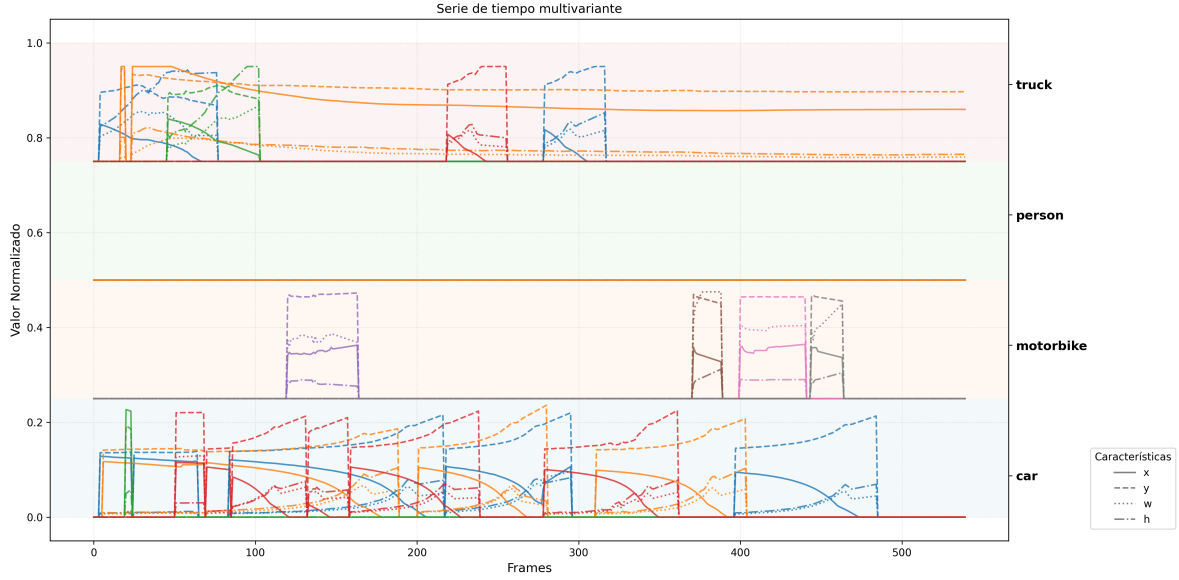


Figura 3.3: Ejemplo de la representación gráfica de una serie de tiempo.

3.4. Mecanismo de ventana deslizante para homologar la longitud de las series de tiempo

A las series de tiempo extraídas se les asigna una etiqueta de clase (*normal*, *suspicious* o *evident*) tomando como referencia el subdirectorío en el que se encuentra el video del cual derivan dentro de los *datasets*.

Como se explica en la especificación del *dataset*, los videos *suspicious* y *evident* se limitaron al intervalo representativo del evento, mientras que los videos *normal* contienen comportamientos continuos y suelen ser más largos. Por ello, las series derivadas tienen longitudes variables, aunque el clasificador requiere que todas posean una longitud constante ℓ_{req} .

En la generación fuera de línea descrita en esta sección, cada fotograma procesado produce una observación de la serie; por tanto, su cantidad de observaciones coincide con la cantidad de fotogramas del video de origen.

Sea $\{X_i\}_{i=1}^N$ un conjunto de series de tiempo multivariantes, donde $X_i \in \mathbb{R}^{\ell_i \times D}$ contiene ℓ_i observaciones y D características. Si el video de origen tiene una tasa f_i en fotogramas por segundo, su duración d_i , en segundos, es

$$d_i = \frac{\ell_i}{f_i}. \quad (4)$$

Los problemas a resolver son los siguientes:

- Las longitudes ℓ_i difieren entre series debido a las distintas duraciones de los eventos segmentados; antes del entrenamiento deben homologarse a una longitud ℓ_{hom} que coincida con la longitud requerida por el clasificador, es decir, $\ell_{\text{hom}} = \ell_{\text{req}}$.

- Las observaciones iniciales y finales pueden carecer de información significativa, es decir, sus D componentes pueden ser iguales a cero.
- Si se elige una longitud fija demasiado grande, las series cortas requerirán un *padding* (relleno con ceros) excesivo.
- Si se elige una longitud fija demasiado pequeña, se perderá información valiosa de las series largas (las cuales son comunes en la clase *normal*).

Para resolver estos problemas, en este trabajo se calcula primero la moda d_{moda} , en segundos, de las duraciones d_i de cada video i del *dataset*:

$$d_{\text{moda}} = \text{moda}(\{d_i\}_{i=1}^N). \quad (5)$$

Después, esta duración se convierte en la longitud homologada ℓ_{hom} , medida en observaciones. Debido a que los videos empleados tienen tasas f_i de 29.97 o 30 fps, se adopta $f_{\text{in}} = 30$ fps como tasa de referencia:

$$\ell_{\text{hom}} = \text{round}(d_{\text{moda}} f_{\text{in}}). \quad (6)$$

Para aplicar esta conversión, f_{in} debe coincidir con las tasas f_i de los videos o aproximarlas dentro de una tolerancia previamente aceptada. En este trabajo se consideraron equivalentes 29.97 (NTSC) y 30 fps debido a su diferencia relativa de aproximadamente 0.1 %; si las diferencias fueran mayores, los videos tendrían que remuestrearse a una tasa común.

El procedimiento consta de dos pasos principales (ver Ecuaciones 7 a 15):

1. **Trimming (recorte):**

Para cada serie X_i se elimina la región inicial y final que no contiene información significativa. Los índices a_i y b_i identifican, respectivamente, la primera y la última observación significativa. La Ecuación 7 conserva, en su orden original, las observaciones comprendidas entre ambos índices y define con ellas la serie recortada X'_i .

$$X'_i = \langle x_{i,t} \rangle_{t=a_i}^{b_i}, \quad 1 \leq a_i \leq b_i \leq \ell_i. \quad (7)$$

La Ecuación 8 calcula la cantidad de observaciones que permanecen después del recorte; se suma una unidad porque los índices a_i y b_i están incluidos.

$$|X'_i| = b_i - a_i + 1. \quad (8)$$

2. **Homogeneización a longitud fija ℓ_{hom} :**

■ **Caso corto ($|X'_i| < \ell_{\text{hom}}$):**

Se aplica un *padding* simétrico con observaciones de relleno formadas por D ceros. Primero, la Ecuación 9 calcula la cantidad total p_i de observaciones que deben añadirse:

$$p_i = \ell_{\text{hom}} - |X'_i|. \quad (9)$$

Capítulo 3. Diseño e implementación de la propuesta de solución

Después, las Ecuaciones 10 y 11 distribuyen esas observaciones entre el inicio y el final de la serie. El uso de piso y techo garantiza que ambas cantidades sumen p_i y que difieran, como máximo, en una observación:

$$|P_{\text{before}}| = \left\lfloor \frac{p_i}{2} \right\rfloor. \quad (10)$$

$$|P_{\text{after}}| = \left\lceil \frac{p_i}{2} \right\rceil. \quad (11)$$

La Ecuación 12 concatena el relleno inicial, la serie recortada y el relleno final, en ese orden:

$$\tilde{X}_i = P_{\text{before}} \parallel X'_i \parallel P_{\text{after}}. \quad (12)$$

Por construcción, la serie resultante tiene la longitud homologada requerida:

$$|\tilde{X}_i| = \ell_{\text{hom}}. \quad (13)$$

■ **Caso largo** ($|X'_i| \geq \ell_{\text{hom}}$):

La serie no se acorta para ajustarla a una sola instancia. En su lugar, la Ecuación 14 genera múltiples segmentos mediante una ventana de longitud ℓ_{hom} que se desplaza S observaciones en cada paso. Para el segmento j , los límites $jS + 1$ y $jS + \ell_{\text{hom}}$ indican, respectivamente, su primera y última observación:

$$\tilde{X}_{i,j} = \langle x'_{i,t} \rangle_{t=jS+1}^{jS+\ell_{\text{hom}}}, \quad j = 0, 1, \dots, m_i. \quad (14)$$

La Ecuación 15 determina el mayor índice m_i cuyo segmento completo cabe dentro de X'_i . Por ello, los índices comienzan en cero y cada serie larga produce $m_i + 1$ segmentos:

$$m_i = \left\lfloor \frac{|X'_i| - \ell_{\text{hom}}}{S} \right\rfloor. \quad (15)$$

De este modo, todos los segmentos tienen la longitud homogénea ℓ_{hom} , y j identifica cada uno de los obtenidos a partir de la serie X'_i . Estos conservan la misma etiqueta de clase que la serie original del i -ésimo video.

Ejemplo: supóngase que la tasa de entrada es $f_{\text{in}} = 30$ fps y que la moda de las duraciones es $d_{\text{moda}} \approx 6.67$ s. Entonces, $\ell_{\text{hom}} = \text{round}(6.67 \times 30) = 200$ observaciones. Si además se elige un paso de $S = 100$ observaciones, para un video de la clase *evident* con una longitud original de 600 observaciones se generan cinco ventanas: la primera comprende las observaciones 1 a 200, la segunda las observaciones 101 a 300, y así sucesivamente hasta la ventana de las observaciones 401 a 600. Cada ventana se convierte en una serie de tiempo que, como todas provienen del mismo video, recibe la misma etiqueta *evident*. El ejemplo se ilustra en la Figura 3.4.

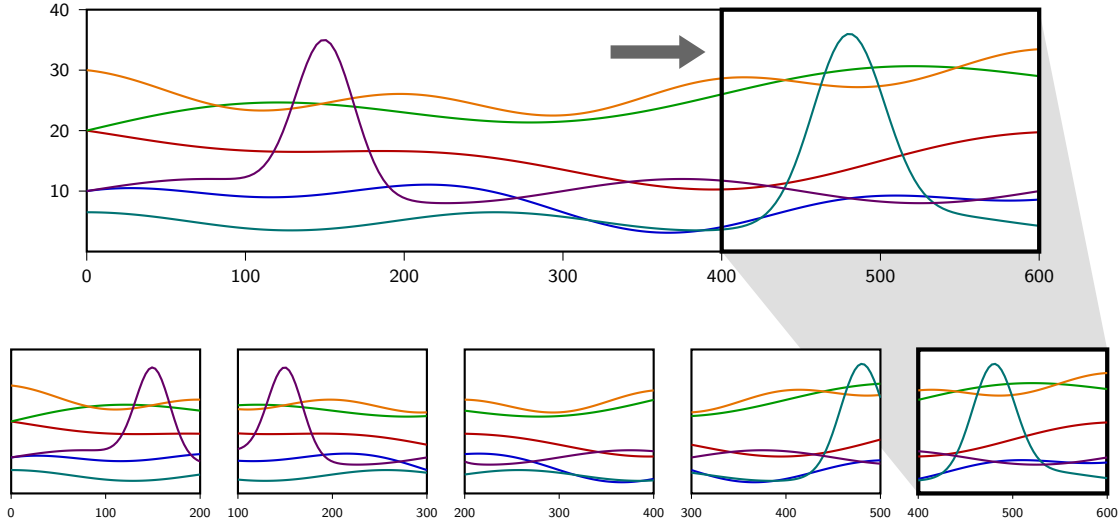


Figura 3.4: Mecanismo de ventana deslizante

Las ventajas de esta técnica, además de homologar la longitud de las series de tiempo a un valor ℓ_{hom} que durante el entrenamiento debe coincidir con la longitud requerida por el clasificador en su capa de entrada (ℓ_{req}), son las siguientes:

- **Aprovecha toda la información útil de los videos largos:** No es necesario recortar cada serie de tiempo por la derecha y/o izquierda descartando información relevante para homologar a una longitud.
- **Reduce el ruido introducido por *padding*:** En lugar de introducir ceros en exceso para intentar homogeneizar la longitud con base en una duración media o máxima (sobre todo en los videos cortos), ℓ_{hom} se determina a partir de una duración representativa cercana al extremo inferior de la distribución, pues las duraciones de los videos presentan un sesgo hacia la derecha.
- **Es consistente con la implementación del *pipeline* de tiempo real:** En este escenario, las inferencias del clasificador de series de tiempo se realizarían periódicamente. Para el ejemplo anterior, sería cada $S/f_{\text{in}} = \frac{100 \text{ f}}{30 \text{ f/s}} = 3.33 \text{ s}$ (segundos) sobre una serie que captura la información de los últimos $\ell_{\text{hom}}/f_{\text{in}} = \frac{200 \text{ f}}{30 \text{ f/s}} = 6.66 \text{ s}$.

Además, este mecanismo representa en sí mismo una técnica de aumento de datos: cada ventana de la misma longitud es una variante válida del mismo evento, solo que desplazado cierta cantidad de tiempo.

Si un video i produce $m_i + 1$ ventanas, entonces la cantidad de muestras de series de tiempo efectivas en el *dataset* aumentado se calcula mediante la Ecuación 16:

$$N_{\text{efectivo}} = \sum_{i=1}^N (m_i + 1) \quad (16)$$

Donde $N_{\text{efectivo}} \geq N$, siendo N_{efectivo} la cantidad de series de tiempo en el *dataset* aumentado, mientras que N representa la cantidad de series de tiempo en el *dataset* original.

Al entrenar con múltiples ventanas correspondientes a un mismo evento, pero desplazadas en el tiempo, se introduce una forma de regularización implícita y se promueve la invarianza temporal. De este modo, el modelo aprende a identificar el evento sin depender de su ubicación específica dentro de la serie de tiempo.

3.5. Mecanismo de permutaciones de *slots* para aumentar el *dataset* de series de tiempo

Un *slot* agrupa la secuencia de variables $\langle x, y, w, h \rangle$ asociadas a un único objeto. Si se tienen 4 clases y se permite rastrear hasta 4 objetos por clase, entonces la serie de tiempo contiene 16 *slots*, organizados en bloques de 4 *slots* por clase.

Entonces, las series de tiempo multivariantes construidas tienen en total una cantidad constante de 64 variables, pero muchas de ellas deben permanecer en cero, ya que la cantidad de agentes presentes en la escena varía entre videos y también a lo largo del tiempo dentro de un mismo video.

Se supuso que esto podría afectar el aprendizaje porque, si ciertos *slots* solían estar activos y otros permanecer en cero, el modelo podría aprender a asociar dichas posiciones con la existencia de objetos, en lugar de aprender las relaciones dinámicas reales entre las variables de movimiento.

Es por ello que se sugirió la técnica de permutaciones de *slots*. Esta técnica consiste en reasignar de forma controlada los valores de un *slot* completo a otro *slot* dentro del bloque de la misma clase, respetando siempre el orden interno de la secuencia $\langle x, y, w, h \rangle$, así como la disposición contigua de todos los *slots* de una misma clase en un bloque y el orden lexicográfico entre los bloques de acuerdo con el nombre de sus clases, en el *dataframe*. El total de intercambios posibles para cada serie de tiempo se calcula a partir de las Ecuaciones 17 y 18.

La cantidad de formas distinguibles de reordenar los *slots* de cada clase se obtiene mediante la fórmula de permutaciones con repetición, simplificada porque únicamente se repite el grupo de elementos nulos. Para cada clase k , con $k \in \{1, \dots, K\}$:

$$\text{Formas}_k = \frac{n_k!}{(n_k - r_k)!} \quad (17)$$

donde:

- k : índice de una de las K clases válidas de objetos.
- n_k : número total de *slots* en el bloque de la clase k .
- r_k : número de *slots* no vacíos y distintos en el bloque de la clase k .
- $n_k - r_k$: número de *slots* vacíos en el bloque de la clase k .

Esto asegura no contar la permutación de los elementos nulos entre sí, ni de diferentes nulos con un mismo no nulo.

Luego, para todas las K clases presentes en una serie de tiempo multivariante:

$$\text{Formas} = \prod_{k=1}^K \text{Formas}_k = \prod_{k=1}^K \frac{n_k!}{(n_k - r_k)!} \quad (18)$$

Por ejemplo, consideremos el caso de una serie de tiempo multivariante cuya cantidad de *slots* por clase es $n_k = A = 4$ constante para todas las clases, teniendo un total de $K = 4$ clases. Si todos sus *slots* fueran no nulos y distintos entre sí, la cantidad de permutaciones posibles sería de $\prod_{k=1}^4 4! = (4!)^4 = 331,776$. Por el contrario, bajo la misma configuración de n_k y K , si dos de las clases tuvieran 3 *slots* no nulos y distintos entre sí, y las otras dos clases tuvieran todos los *slots* nulos, la cantidad de permutaciones sería de $\left(\frac{4!}{1!}\right)^2 \cdot \left(\frac{4!}{4!}\right)^2 = 24^2 \cdot 1 = 576$.

Como se puede observar, la cantidad de permutaciones posibles es muy alta, y por esa razón al ejecutar el experimento, se elige aleatoriamente solo un subconjunto de todas las permutaciones posibles.

3.6. Integración de OD, MOT y TSC en *pipeline* para la inferencia en tiempo real

Para garantizar la operación del sistema en tiempo real y evitar la acumulación de retrasos, se diseñó una arquitectura de *software* basada en un patrón productor-consumidor.

Denotamos t_{gen} al tiempo promedio que demora procesar cada fotograma en los módulos OD y MOT hasta construir una observación (muestra) de la serie de tiempo multivariante. Asimismo, $f_{\text{in}} = 1/t_{\text{in}}$ representa la tasa de llegada de fotogramas (en fps) determinada por el flujo de video de entrada desde la cámara.

Dado que el tiempo requerido para extraer las características espaciales de un fotograma (t_{gen}) suele ser significativamente mayor que el tiempo de entrega de un fotograma desde la cámara o video (t_{in}), e incluso presenta variaciones (incrementándose a medida que el MOT debe rastrear una mayor cantidad de objetos), un enfoque secuencial provocaría el bloqueo del flujo de video, haciendo que no fuera en tiempo real.

Para resolver esto, el proceso se desacopló en tres hilos de ejecución concurrentes, descritos a continuación:

1. Hilo de Captura de Video (Productor Asíncrono)

- **Naturaleza de ejecución:** Opera de manera asíncrona e independiente respecto al resto del sistema, garantizando que el flujo de entrada nunca se bloquee por la carga computacional posterior.
- **Proceso:** Su función exclusiva es leer fotogramas de la cámara a su frecuencia de origen (f_{in}). Para satisfacer las restricciones operativas de tiempo real dictadas por t_{gen} , este hilo implementa el mecanismo de *downsampling*. Al descartar fotogramas estratégicamente, reduce la tasa de entrada para producir una tasa de fotogramas efectiva (f_{eff}). Esta tasa se establece introduciendo el parámetro de configuración del sistema N_{skip} , el cual define el número de fotogramas omitidos secuencialmente, tal como se expresa en la Ecuación 19:

$$f_{\text{eff}} = \frac{f_{\text{in}}}{1 + N_{\text{skip}}} \quad (19)$$

El valor resultante de f_{eff} debe cumplir con dos criterios. Primero, el criterio de Nyquist-Shannon (Shannon, 1949), definido como $f_{\text{eff}} \geq 2 \cdot f_{\text{max}}$, donde f_{max} representa la frecuencia máxima relevante de la dinámica del movimiento, descartando altas frecuencias atribuibles al ruido. Segundo, debe respetar la capacidad límite de procesamiento del sistema, de manera que se cumpla que $f_{\text{eff}} \leq 1/t_{\text{gen}}$.

- **Salida:** Los fotogramas seleccionados se depositan en una cola asíncrona (*queue*) de memoria compartida, listos para ser consumidos por el hilo de procesamiento espacial. Para mitigar los cuellos de botella transitorios —es decir, cuando t_{gen} experimenta picos debido a una alta densidad de objetos en escena y vulnera temporalmente la condición $f_{\text{eff}} \leq 1/t_{\text{gen}}$ — esta cola se diseña con una capacidad máxima unitaria. En consecuencia, si el hilo de procesamiento se encuentra ocupado resolviendo un fotograma anterior, los nuevos fotogramas entrantes son descartados activamente (*frame dropping*). Este mecanismo garantiza que el sistema siempre evalúe la observación espacial más reciente disponible, evitando la acumulación de latencia en la predicción.

2. Hilo de Procesamiento Espacial (Consumidor/Productor Asíncrono)

- **Naturaleza de ejecución:** Actúa de forma síncrona respecto a la cola de fotogramas (procesa uno tras otro conforme llegan), pero de forma asíncrona respecto a la captura de la cámara y a las inferencias del clasificador temporal.
- **Proceso:** Por cada fotograma extraído de la cola, ejecuta la fase de visión computacional. Primero, el módulo OD ubica los elementos de interés. Inmediatamente después, pasa las detecciones al módulo MOT, que asocia un identificador único a cada trayectoria a lo largo del tiempo y extrae las coordenadas espaciales normalizadas.
- **Salida:** El resultado se inserta en un búfer (*buffer*) circular de tamaño fijo. Cada inserción empaqueta una marca de tiempo absoluta (*timestamp*) del sistema operativo y un vector que representa el estado espacial de todos los objetos rastreados en ese instante de tiempo.

3. Hilo de Clasificación Temporal (Consumidor Asíncrono)

- **Naturaleza de ejecución:** Opera de manera completamente asíncrona respecto al flujo de video. Su ejecución está gobernada por un temporizador independiente que se dispara estrictamente cada T_i segundos, siendo éste un parámetro de entrada del sistema configurable, el cual se debe establecer de manera tal que T_i sea mayor o igual al tiempo requerido para la inferencia t_{cls} del modelo TSC utilizado.
- **Proceso:** Este hilo consolida la información para el análisis temporal mediante tres pasos secuenciales:
 - *Extracción:* Consulta el búfer circular y extrae únicamente los estados espaciales cuyas marcas de tiempo absolutas (*timestamps*) se encuentren dentro de la ventana de los últimos T_w segundos, garantizando que no exista desalineación temporal con la realidad, sin importar si el sistema procesó más o menos fotogramas en ese lapso. T_w es también un parámetro de entrada del sistema.
 - *Alineación e Interpolación:* Dado que el tiempo de procesamiento t_{gen} fluctúa y se aplica *downsampling*, la cantidad de muestras almacenadas en el periodo T_w es variable. Como el modelo clasificador exige una entrada temporal de longitud fija ℓ_{req} , el sistema aplica un remuestreo mediante interpolación. Esto proyecta la serie temporal de longitud variable a exactamente ℓ_{req} pasos temporales discretos, homogeneizando la señal.
 - *Clasificación:* El tensor multivariante resultante ingresa al módulo TSC, que infiere el comportamiento del segmento observado en el periodo T_w .

- **Salida:** El sistema reporta la etiqueta predicha (ej. *normal*, *suspicious*, *evident*) y actualiza el temporizador para el siguiente ciclo T_i . En otras palabras, emite una inferencia cada T_i segundos sobre una serie de tiempo multivariante de longitud ℓ_{req} que (después del tratamiento por interpolación y estampas de tiempo real) representa de forma fiel lo observado en los últimos T_w segundos reales.

La Figura 3.5 ilustra el *pipeline* asíncrono productor-consumidor compuesto por tres hilos de ejecución concurrentes. En el primer hilo se representa la captura a $f_{\text{in}} = 30$ fps, donde el *downsampling* con $N_{\text{skip}} = 2$ se visualiza mediante los fotogramas grises, los cuales son omitidos para que solo uno de cada tres sea candidato a procesamiento. Los fotogramas verdes corresponden a los que logran pasar al segundo hilo, mientras que los rojos representan descartes (*drops*) producidos por saturación de la cola cuando OD+MOT no consume los datos con suficiente rapidez.

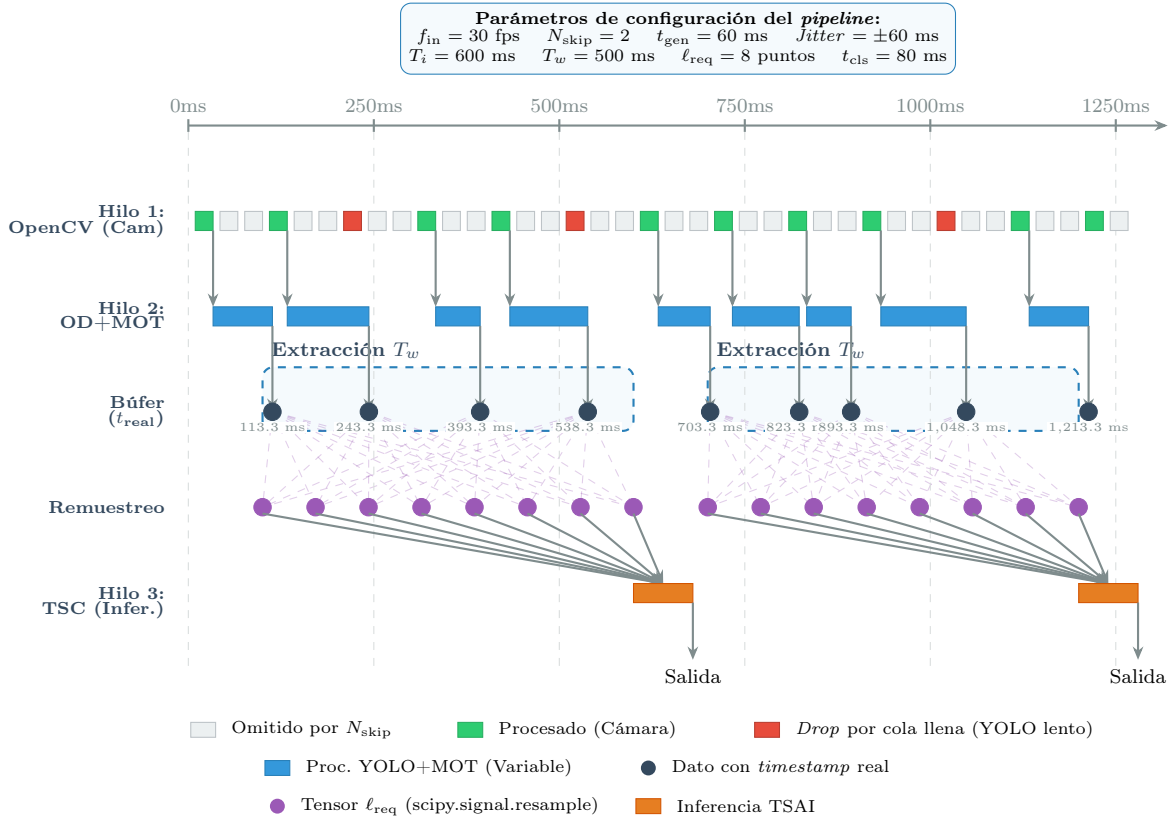


Figura 3.5: Representación temporal del *pipeline* asíncrono productor-consumidor

En el segundo hilo, los rectángulos azules representan la ejecución de OD+MOT y su longitud indica el tiempo variable de procesamiento asociado al *jitter*. Por esta razón, las salidas no llegan al búfer con separación uniforme, sino con marcas de tiempo reales t_{real} , mostradas como puntos oscuros. Así, aun con una tasa de captura constante y *downsampling*, la secuencia disponible para clasificación puede contener retardos, pérdidas e irregularidades temporales.

El tercer hilo ejecuta la inferencia TSC de forma periódica cada T_i , independientemente del estado de los hilos anteriores. En cada activación, toma retrospectivamente del búfer los datos disponibles dentro

de una ventana T_w , representada por las regiones punteadas, y los remuestrea para obtener un tensor de longitud fija ℓ_{req} . Los puntos morados muestran las muestras interpoladas usadas como entrada del modelo. Finalmente, los rectángulos naranjas representan cada inferencia TSC: su posición inicial indica el instante de activación y su anchura corresponde al tiempo de clasificación $t_{\text{cls}} = 80$ ms, tras el cual se obtiene la predicción del modelo.

Cabe destacar que existe una regla estricta entre la cantidad de muestras que exige la red neuronal (ℓ_{req}) y los segundos reales de video analizados (T_w). Durante la configuración de este *software* no debe modificarse arbitrariamente la relación ℓ_{req}/T_w , ya que este cociente debe coincidir con la frecuencia de muestreo de las series de tiempo con las que se entrenó el modelo TSC.

Por ejemplo, supóngase que el entrenamiento se realizó con series de tiempo generadas a partir de videos de 30 fps, construyendo ventanas de $\ell_{\text{req}} = 150$ muestras que representan $T_w = 5$ segundos de tiempo real.

En primer lugar, ℓ_{req} no puede alterarse, ya que está fijado por la arquitectura de la capa de entrada del modelo TSC entrenado. Cualquier modificación en este valor provocaría un error de dimensionalidad durante la inferencia.

Por otro lado, dado que ℓ_{req} debe permanecer fijo (150 en el ejemplo), cambiar T_w a un valor distinto de 5 segundos de tiempo real no produciría dicho error, pero sí alteraría el cociente ℓ_{req}/T_w . En términos prácticos, esto equivale a modificar la escala temporal de la señal, comprimiendo o estirando artificialmente la velocidad física de los objetos rastreados, lo cual invalidaría las predicciones del modelo.

Por lo tanto, si se requiere operar con una ventana temporal T_w diferente y, en consecuencia, con otra resolución temporal, es necesario regenerar el conjunto de datos de series de tiempo y volver a entrenar el modelo TSC utilizando la nueva frecuencia de muestreo ℓ_{req}/T_w , ya sea haciendo submuestreo o interpolación sobre las series de tiempo o empleando videos con la tasa de fotogramas adecuada, manteniendo el tamaño de ventana ℓ_{req} que se desee para representar el intervalo temporal correspondiente. También se puede considerar un mecanismo de aumento de datos de ajuste de escala de tiempo, pero bajo rangos controlados para permitir al modelo ser resiliente ante pequeños cambios de escala temporal.

3.7. Implementación por computadora de la propuesta de solución

Tal como se mostró en el diagrama de la Figura 3.1, se implementaron tres herramientas de *software*, todas ellas programadas en lenguaje Python. A continuación, se resume la implementación de cada una de ellas.

3.7.1. Herramienta para la generación de *datasets* de series de tiempo

Antes de iniciar el programa, el usuario puede configurar un archivo JSON5 con un conjunto de parámetros que permiten especificar:

- El archivo de pesos del modelo de detección de objetos (formatos *.pt* u *.onnx*).

- El subsistema MOT a utilizar (DeepSORT, ByteTrack o Deep OC-SORT), junto con sus parámetros de configuración.
- La cantidad de *slots* deseada por clase de objeto. Actualmente solo es posible asignar la misma cantidad para todas las clases (A), es decir, no se puede especificar un número distinto para cada clase (*car*, *motorcycle*, *person*, *truck*).
- La ventana máxima (W) y el porcentaje de estabilidad (ρ) empleados en el Algoritmo 1 para la estabilización de la clase de un objeto.
- Las áreas mínimas por clase de objeto a partir de las cuales una detección del subsistema OD se considera válida para su incorporación al MOT.
- La longitud homologada de la ventana de la serie de tiempo (ℓ_{hom}).
- El tamaño de paso (S), que puede establecerse de forma independiente para cada clase de evento (*evident*, *normal*, *suspicious*).
- Una nota descriptiva del experimento a realizar.

La generación de un *dataset* de series de tiempo multivariantes sigue una secuencia de pasos iniciados por el usuario a través de las opciones mostradas en consola:

1. **Generación de series de tiempo individuales:** Para cada video, se genera su serie de tiempo correspondiente, la cual se almacena como un *dataframe* en formato CSV, junto con su representación gráfica en formato PNG.
2. **Construcción del *dataset* etiquetado:** Los archivos CSV generados en el paso anterior se integran en un único *dataset* etiquetado, también almacenado en formato CSV.
3. **Homogeneización de la dimensión temporal:** La longitud de las series de tiempo se ajusta mediante un esquema de ventana deslizante, de modo que todas tengan la misma cantidad de muestras (observaciones). El resultado se almacena en un nuevo archivo CSV. Para un mismo *dataset* etiquetado, es posible generar múltiples variantes homogenizadas con distintas configuraciones de ventana y paso.
4. **Aumento de datos:** El tamaño del *dataset* se incrementa aplicando el mecanismo de permutación de *slots* sobre las series de tiempo. El resultado se almacena en un archivo CSV independiente de los anteriores.

Además, se tienen las siguientes opciones de diagnóstico:

- **Análisis de la duración de videos de un *dataset* de videos:** Genera estadísticas descriptivas sobre la cantidad de *frames* por video, tanto a nivel global como por clase.
- **Análisis del rendimiento del procesamiento:** Genera gráficos que relacionan la duración de los videos con el tiempo requerido para procesarlos, a partir de métricas capturadas durante la generación de las series de tiempo.

Cada ejecución genera una carpeta de experimento diferente con la estructura que se presenta en la Figura 3.6.

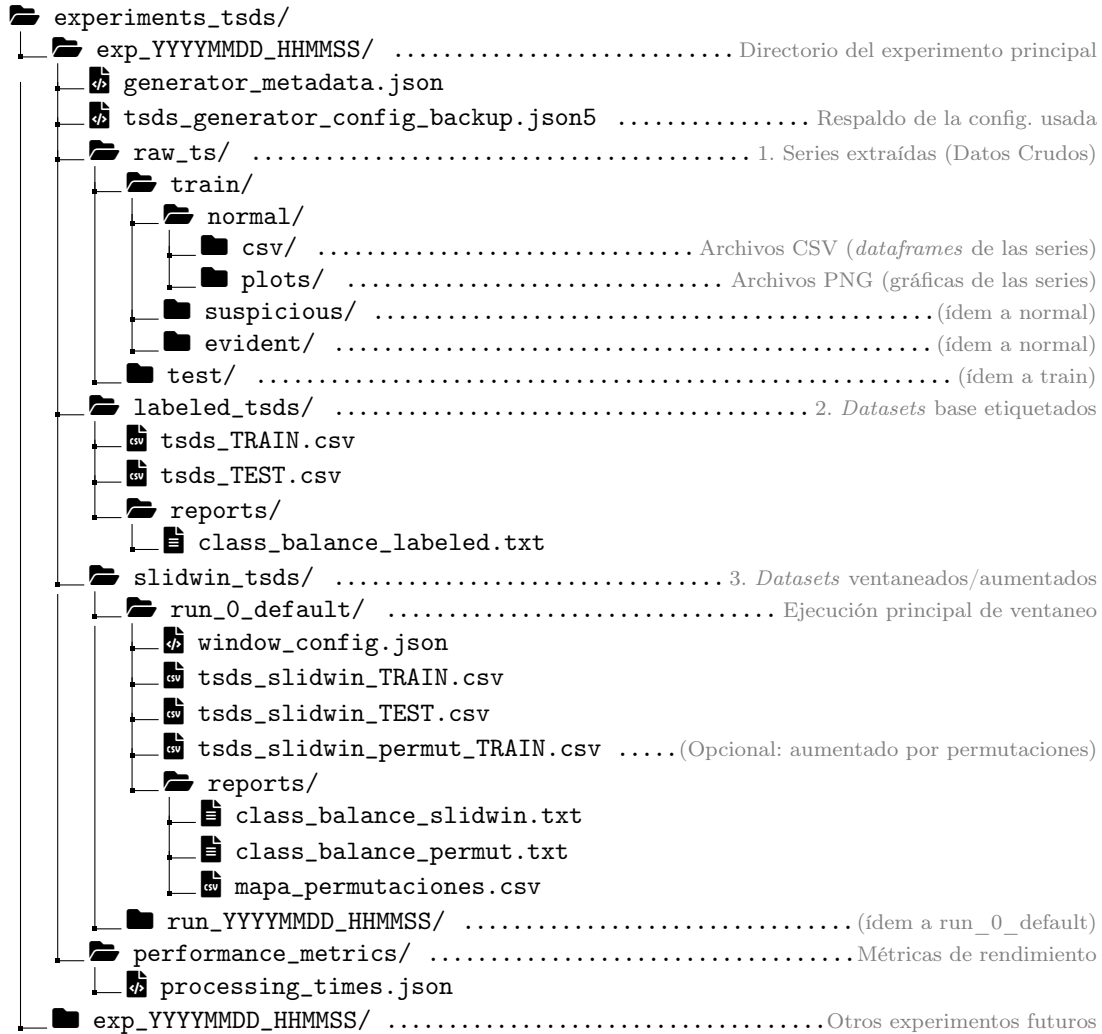


Figura 3.6: Árbol de directorios de los archivos generados por la herramienta para la generación de *datasets* de series de tiempo.

3.7.2. Herramienta para el entrenamiento de modelos de clasificación de series de tiempo

Con este *software*, previa configuración de un archivo JSON, es posible realizar las siguientes actividades:

1. Entrenamiento de modelos de clasificación de series de tiempo

Permite seleccionar arquitecturas disponibles en la biblioteca de código TSAI y utilizar *datasets* de distinta naturaleza:

- *Datasets* públicos del repositorio web UCR accesibles mediante la API de TSAI.
- *Datasets* almacenados en archivos locales en formatos ARFF, CSV o TS.
- *Datasets* en formato CSV generados mediante la herramienta descrita en la Sección 3.7.1.

En este caso, se tienen controles específicos para evitar la contaminación de datos (*data leakage*): se genera una lista de grupos que puede utilizar *StratifiedGroupKFold* de *scikit-learn* para que todas las ventanas de series de tiempo derivadas de un mismo video se asignen exclusivamente al conjunto de entrenamiento o al de prueba, pero no a ambos durante la validación cruzada. Además, se evita que series de tiempo provenientes de videos sintéticos sean incluidas en el conjunto de prueba.

Mediante la interfaz de línea de comandos (CLI) es posible seleccionar directamente cada uno de los *datasets* generados por la herramienta presentada en la Sección 3.7.1.

2. Configuración de estrategias de experimentación

Permite definir distintos aspectos del proceso de entrenamiento:

- Selección del esquema de evaluación: *splits* originales o validación cruzada, indicando el número de repeticiones.
- Definición de estrategias de normalización de las series de tiempo (aplicar o no normalización antes del entrenamiento y decidir si normalizar por serie, por variable o por observación).
- Selección del optimizador (Adam, AdamW, AdaBelief o RMSProp).
- Elección de la función de pérdida (por defecto o *weighted cross-entropy*).
- Configuración de hiperparámetros de entrenamiento: número de épocas, habilitación de paro temprano (*early-stopping*, especificando paciencia), tamaño de *batch* y tasa de aprendizaje.
- De manera opcional, mediante un segundo archivo JSON5, definición de hiperparámetros específicos de cada arquitectura (tamaño de *kernel*, habilitación de ramas, etc.).

3. Generación de reportes de balance de clases y *splits*

Permite analizar el balance entre las clases y los subconjuntos de entrenamiento y pruebas, así como el resultado de una validación automática que descarta la existencia de *data leakage*. Estos reportes se generan por cada *split* generado en la validación cruzada.

4. Generación de reportes de métricas de rendimiento

Permite evaluar los modelos entrenados en métricas como *accuracy*, *macro average precision*, *macro average recall* (*balanced accuracy*), *macro f1-score*, *loss*, y tiempo de entrenamiento. Asimismo, se generan y almacenan gráficas como la matriz de confusión, curvas de aprendizaje y pérdida, y muestras de *batches*.

5. Almacenamiento de modelos

Se guardan los modelos con menor *loss* en el conjunto de prueba.

6. Medición del tiempo de inferencia

Se mide el tiempo que le toma a cada modelo entrenado hacer la inferencia por cada serie de tiempo.

7. Análisis estadístico y generación de *rankings*

Permite generar *rankings* de clasificadores a partir de una métrica seleccionada, aplicando pruebas estadísticas no paramétricas como el test de Friedman y el test de rangos con signo de Wilcoxon con corrección de Holm, así como la generación de diagramas de diferencias críticas para identificar grupos de modelos sin diferencias estadísticamente significativas. Esto es útil cuando se evalúan varios modelos a través de múltiples *datasets*.

Capítulo 3. Diseño e implementación de la propuesta de solución

Cada ejecución genera una carpeta de experimento diferente con la estructura que se presenta en la Figura 3.7.

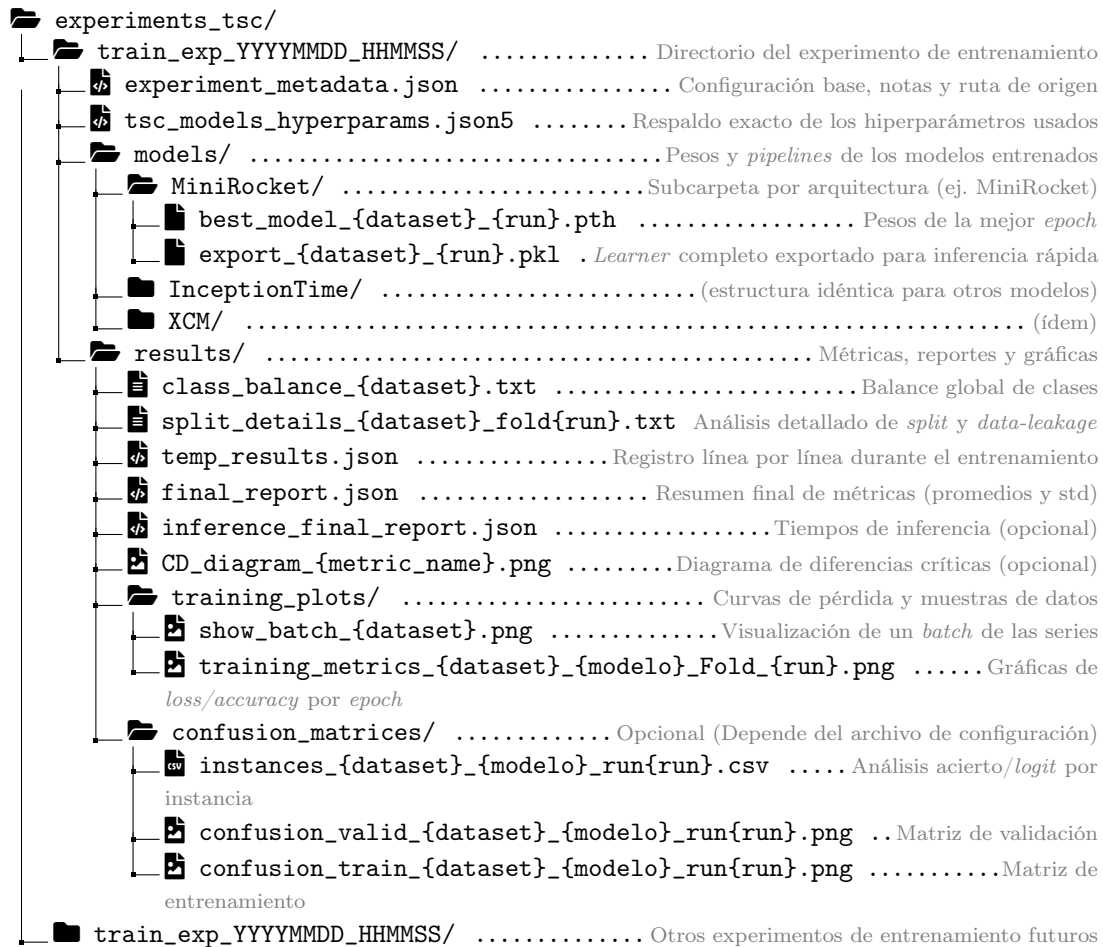


Figura 3.7: Árbol de directorios de los archivos generados por la herramienta para el entrenamiento de modelos TSC.

3.7.3. Herramienta para la prueba del *pipeline* en tiempo real

Con esta herramienta es posible capturar video desde una cámara mediante OpenCV y ejecutar el *pipeline* de OD, MOT y TSC. Mediante el uso de OBS Studio es posible generar una cámara virtual que apunte a una ventana del reproductor multimedia que reproduzca un video de prueba, con el fin de que OpenCV la capture como si se tratara de una cámara real.

La interfaz gráfica (ver Figura 3.8) incluye una ventana que muestra el video en tiempo real. Para evitar la acumulación de retrasos, se emplea un esquema asíncrono productor-consumidor. Esto implica que el procesamiento y la visualización no necesariamente ocurren a la misma tasa de fotogramas que el video original; sin embargo, los fotogramas mostrados corresponden siempre al instante actual de la señal de entrada. En esta ventana se visualizan los *bounding boxes* de los objetos rastreados por el MOT, utilizando un color distinto según su clase. Esto permite evaluar cualitativamente la precisión

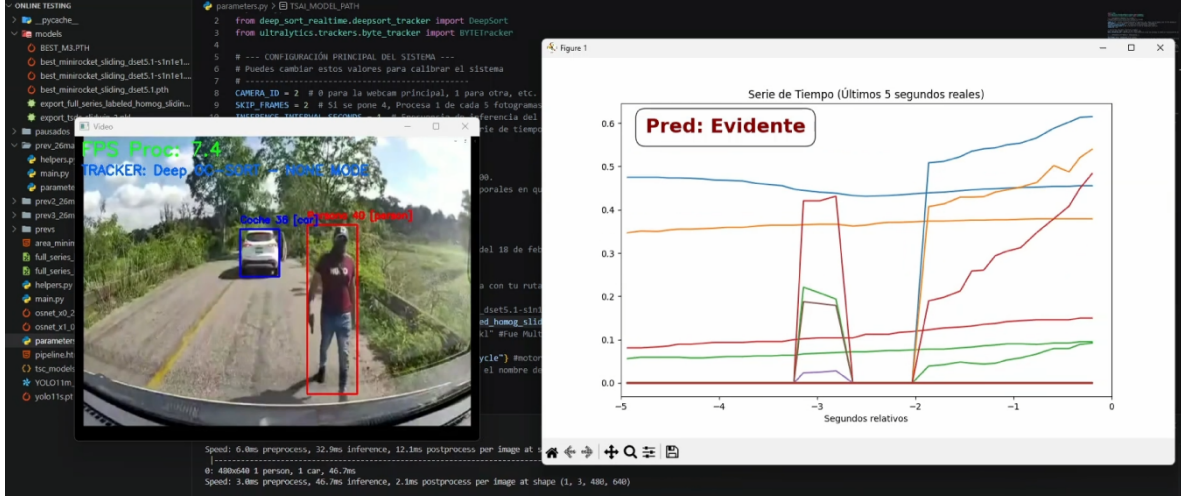


Figura 3.8: Interfaz gráfica de la herramienta para la prueba del *pipeline* en tiempo real.

de las detecciones y la consistencia del seguimiento de trayectorias. Mediante la tecla *d*, es posible alternar a un modo en el que se muestran únicamente las detecciones en crudo del módulo OD, sin pasar por el MOT.

En una segunda ventana (GUI) se muestra la serie de tiempo extraída en tiempo real, sincronizada con el instante observado en la primera ventana (GUI). El tiempo presente se fija en $t = 0$, y se visualiza un segmento temporal¹¹ retrospectivo con la longitud configurada, la cual es precisamente la entrada del TSC. En la parte superior de la gráfica se indica la etiqueta inferida para dicho segmento. Las series de tiempo representadas corresponden a los valores reales utilizados por el TSC, sin aplicar escalamiento ni separación por bandas según la clase de objeto.

3.8. Detalles prácticos sobre la implementación

Las tres herramientas se programaron en lenguaje Python 3.12.8 (64 bits). Como parte de la experimentación se probaron:

- Dos modelos de detección de objetos (YOLO11s y YOLO11m), operados mediante el *framework* *ultralytics* empleando pesos preentrenados en los conjuntos de datos COCO y BDD100K, respectivamente.
- Tres sistemas MOT (Deep OC-SORT, DeepSORT y ByteTrack), implementados mediante las bibliotecas externas *boxmot*, *deep_sort_realtime* y *ultralytics.trackers*, respectivamente.
- Nueve arquitecturas para la clasificación de series de tiempo (HydraMultiRocket, Hydra, InceptionTime, MiniRocket, MultiRocket, OmniScaleCNN, TSPerceiver, TSSequencer y XCM), implementadas a través de la biblioteca *tsai*.

El desarrollo de las herramientas se apoyó extensivamente en diversas bibliotecas estándar básicas de Python para tareas fundamentales de control de flujo, manejo de subprocesos, operaciones matemáticas

¹¹El término *segmento temporal* se utiliza como sinónimo de *ventana temporal* de la serie de tiempo. Se adopta este término para evitar confusión con las ventanas de la interfaz gráfica de usuario (GUI).

Capítulo 3. Diseño e implementación de la propuesta de solución

y manipulación de archivos. Entre estas se incluyen: `argparse`, `collections`, `datetime`, `gc`, `glob`, `itertools`, `json`, `math`, `os`, `pathlib`, `random`, `re`, `shutil`, `sys`, `traceback` y `typing`.

Sin embargo, las bibliotecas de código más importantes y determinantes en el funcionamiento principal de cada herramienta se resumen en la Tabla 3.1. Entre ellas se encuentran la biblioteca `sklearn` (particularmente el módulo `sklearn.metrics`) y `time`, cuyas funciones se utilizaron para implementar las métricas de evaluación.

Tabla 3.1: Bibliotecas principales utilizadas en el desarrollo de las herramientas.

Biblioteca (Nombre oficial)	TSDS Generator ^a	TSC Evaluator ^b	RT ^c
<code>adabelief_pytorch</code> ¹²		✓	
<code>aeon.visualisation</code> ¹³		✓	
<code>boxmot</code> ¹⁴	✓		✓
<code>cv2</code> (OpenCV)	✓		✓
<code>deep_sort_realtime</code> ¹⁵	✓		✓
<code>fastai</code>		✓	
<code>json5</code> ¹⁶	✓	✓	✓
<code>matplotlib</code>	✓	✓	✓
<code>numpy</code>	✓	✓	✓
<code>pandas</code>	✓	✓	✓
<code>queue</code>			✓
<code>scipy</code>		✓	✓
<code>seaborn</code>	✓		
<code>sklearn</code> (scikit-learn)		✓	✓
<code>threading</code>			✓
<code>time</code>	✓	✓	✓
<code>torch</code> (PyTorch)	✓	✓	✓
<code>torchvision</code>	✓		✓
<code>tsai</code> ¹⁷		✓	✓
<code>ultralytics</code>	✓		✓
<code>ultralytics.trackers</code>	✓		✓

Notas:

^a **TSDS Generator:** Herramienta para la generación de *datasets* de series de tiempo, descrita en la sección 3.7.1.

^b **TSC Evaluator:** Herramienta para el entrenamiento de modelos de clasificación de series de tiempo, descrita en la sección 3.7.2.

^c **RT:** Herramienta para la prueba de *pipeline* en tiempo real, descrita en la sección 3.7.3.

¹²Repositorio oficial: <https://github.com/juntang-zhuang/Adabelief-Optimizer>

¹³Repositorio oficial: <https://github.com/aeon-toolkit/aeon>

¹⁴Repositorio oficial: <https://github.com/mikel-brostrom/boxmot>

¹⁵Repositorio oficial: https://github.com/levan92/deep_sort_realtime

¹⁶Repositorio oficial: <https://github.com/Kijewski/pyjson5>

¹⁷Repositorio oficial: <https://github.com/timeseriesAI/tsai>

Capítulo 4

Experimentación y Resultados

Durante el desarrollo de este trabajo, se han ejecutado cerca de cincuenta pruebas, lo que ha facilitado la mejora iterativa de las propias herramientas.

En el análisis de las duraciones de los eventos se obtuvo una moda de 5.4 segundos, equivalente a 162 observaciones a 30 fps. Para los experimentos, esta duración se redondeó por practicidad a 5 segundos y, en consecuencia, se estableció $\ell_{\text{hom}} = \ell_{\text{req}} = 150$ observaciones. Los videos reales presentan tasas de 29.97 o 30 fps, cuya diferencia relativa es aproximadamente 0.1 %. Por tanto, ambas se procesaron tomando 30 fps como referencia: una ventana de 150 observaciones abarca aproximadamente 5.005 segundos a 29.97 fps y 5 segundos a 30 fps. Esta diferencia, cercana a 5 ms, se consideró tolerable y no requirió remuestreo adicional.

Específicamente, se realizaron pruebas combinando dos modelos de detección de objetos (OD) —YOLO11s entrenado en el *dataset* COCO y YOLO11m entrenado en el *dataset* BDD100K— y tres sistemas de seguimiento múltiple de objetos (MOT): DeepSORT, ByteTrack y Deep OC-SORT. Para ello, se aplicaron dos metodologías de evaluación cualitativas y analíticas:

1. **Análisis de la representación gráfica de las series de tiempo:** Se contrastaron cualitativamente las gráficas generadas (como la de la Figura 3.3) por la herramienta para la generación de *datasets* de series de tiempo, descrita en las secciones 3.2 y 3.3 con los videos, para verificar que representaran fielmente el movimiento de los agentes principales. De estas observaciones se derivaron los siguientes ajustes y conclusiones:
 - **Ajuste de hiperparámetros MOT:** Se configuraron los parámetros de los sistemas MOT para favorecer la continuidad de las variables de la serie de tiempo y reducir las caídas abruptas a cero causadas por pérdidas temporales de seguimiento.
 - **Mitigación de *flickering*:** Se identificó el problema de fluctuación rápida (*flickering*) en las clases de los objetos detectados, que se solucionó mediante la implementación del mecanismo de estabilización descrito en el Algoritmo 1.
 - **Importancia del módulo ReID:** Aunque la literatura reporta que ByteTrack supera el rendimiento de DeepSORT prescindiendo de un módulo de reidentificación (ReID), en la práctica y tras probar múltiples configuraciones, no logró igualar la continuidad de las trayectorias de DeepSORT ni de Deep OC-SORT. Esto sugiere que, en escenarios de carretera, el uso de ReID es clave.

2. **Monitoreo de *bounding boxes* en tiempo real:** Se evaluó visualmente el ajuste de los *bounding boxes* sobre los vehículos utilizando la herramienta de pruebas del *pipeline* en tiempo real descrita en la sección 3.7.3. Los principales hallazgos y soluciones implementadas fueron:

- **Adopción de Deep OC-SORT:** La configuración de DeepSORT, orientada a mantener continuidad, junto con su Filtro de Kalman, tendía a extrapolar trayectorias pasadas e ignorar cambios bruscos o aceleraciones reales. Por ello se optó por Deep OC-SORT, que mantiene detecciones prolongadas pero con mejor respuesta a movimientos abruptos.
- **Filtro geométrico para falsos positivos:** Se detectó que YOLO11s (COCO) generaba abundantes falsos positivos al clasificar el cofre del propio ego-vehículo como otro automóvil. Esto se resolvió implementando el filtro geométrico descrito en la propuesta de solución.
- **Compromiso (*trade-off*) entre modelos OD:** Se observó que YOLO11s (COCO) presentaba bajo rendimiento en condiciones de poca iluminación o adversas. Aunque YOLO11m (BDD100K) reduce esta deficiencia para los vehículos, presenta una mayor tasa de fallos en la detección de personas.

Para llevar estas evaluaciones a un plano cuantitativo, sería indispensable definir métricas de calidad para las series de tiempo. Estas podrían calcularse de manera intrínseca (midiendo, por ejemplo, la tasa de caídas abruptas a cero sin depender de información externa) o mediante la integración de un *ground truth* cinemático dentro del *dataset* de robo a transporte de carga. La definición e implementación de estas métricas queda fuera del alcance de esta investigación y se plantea como trabajo futuro.

También se realizó un *benchmark* de nueve arquitecturas TSC de la biblioteca TSAI, entrenadas con catorce *datasets* de series de tiempo multivariantes del repositorio público UEA. Cada arquitectura se probó con ocho configuraciones experimentales, obtenidas al combinar cuatro optimizadores (Adam, AdamW, AdaBelief y RMSProp) con dos opciones de preprocesamiento: aplicar o no normalización Z antes del entrenamiento. Para estas configuraciones se midieron el rendimiento y los tiempos de entrenamiento e inferencia. Posteriormente, se elaboró un *ranking* y se aplicaron pruebas estadísticas no paramétricas basadas en los *tests* de Friedman y Wilcoxon-Holm. Los resultados están publicados en el capítulo de libro de Gutiérrez Montor et al. (2027). Este estudio fue importante para seleccionar las arquitecturas TSC utilizadas en los experimentos que se presentan a continuación.

4.1. Experimento 1

4.1.1. Diseño experimental

- **Objetivo:** Evaluar el desempeño y la capacidad de generalización de arquitecturas recientes de clasificación de series de tiempo en el reconocimiento de patrones normales, sospechosos y evidentes de robo a transporte de carga, así como analizar el impacto de la filtración de datos (*data leakage*) al contrastar dos métodos de validación cruzada.
- **Tipo de diseño:** Diseño cuasiexperimental de corte transversal.
- **Objetos:** Se evaluaron nueve arquitecturas de clasificación de series de tiempo (HydraMultiRocket, Hydra, InceptionTime, MiniRocket, MultiRocket, OmniScaleCNN, TSPerceiver, TSSequencer y XCM). El *dataset* evaluado corresponde a series de tiempo generadas sin aumento de datos (clave de control TSDS1).
- **Instrumentos:**

Para evaluar cada una de las arquitecturas en las métricas seleccionadas, se utilizó la herramienta para el entrenamiento de modelos de clasificación de series de tiempo, descrita en la sección 3.7.2.

Para ejecutar esta herramienta, se utilizó el equipo de cómputo descrito en la Tabla 4.1.

Tabla 4.1: Equipo de cómputo utilizado para la ejecución de los experimentos

Componente	Especificación
CPU	AMD Ryzen 7 5700X (8 núcleos a 3.4 GHz)
RAM	16 GB a 2400 MT/s
GPU	NVIDIA GeForce RTX 4060
VRAM	8 GB
Sistema operativo	Windows 11 Pro 24H2

- **Proceso de evaluación y métricas:** La métrica principal seleccionada para evaluar los modelos fue la exactitud balanceada (*balanced accuracy*) en el conjunto de prueba, la cual es equivalente al *macro average recall* en clasificación multiclase. Su valor se obtiene mediante la ecuación 20

$$Balanced Accuracy = \frac{1}{C} \sum_{k=1}^C Recall_k = \frac{1}{C} \sum_{k=1}^C \frac{TP_k}{TP_k + FN_k} \quad (20)$$

Donde C es la cantidad de clases; TP_k y FN_k corresponden a los verdaderos positivos y falsos negativos de la clase k , respectivamente.

Se eligió esta métrica porque mitiga el impacto del desbalance de clases presente en el *dataset* (Opitz, 2024), en donde la clase “*Normal*” está sobrerrepresentada al provenir de videos de mayor duración.

Para la evaluación se compararon empíricamente dos esquemas de validación cruzada: *Stratified 5-Fold* y *StratifiedGroup 5-Fold*. El uso del esquema *StratifiedGroup* fue estrictamente necesario para aislar la filtración de datos derivada del uso de ventanas deslizantes. Este agrupamiento asegura que todas las ventanas extraídas de un mismo video permanezcan juntas en el conjunto de entrenamiento o en el de prueba, permitiendo medir la verdadera capacidad de generalización del modelo en lugar de su capacidad de memorización.

4.1.2. Ejecución del experimento

- **Dataset:** El *dataset* utilizado en este experimento (clave TSDS1) se construyó procesando el *dataset* de videos reales en su versión 4, grabados a tasas de 29.97 o 30 *fps*. Para extraer las series de tiempo, se utilizaron en pasos previos el modelo YOLO11s (entrenado en COCO) para la detección de objetos y DeepSORT para el seguimiento. Posteriormente, se extrajeron las características cinemáticas aplicando un esquema de ventana deslizante con un tamaño de 5 segundos ($\ell_{\text{hom}} = \ell_{\text{req}} = 150$ observaciones) y un paso temporal de 2 segundos. Este proceso arrojó un total de 2,818 segmentos. La distribución exacta de los subconjuntos de entrenamiento y prueba se detalla en la Tabla 4.2.

Tabla 4.2: Distribución de las clases y subconjuntos del *dataset* TSDS1 (sin aumento de datos)

Clase	Conjunto de entrenamiento ($\approx 80\%$)	Conjunto de prueba ($\approx 20\%$)
<i>Evident</i>	447 (19.83 %)	112 (19.85 %)
<i>Normal</i>	1481 (65.70 %)	370 (65.60 %)
<i>Suspicious</i>	326 (14.46 %)	82 (14.53 %)

- **Entrenamiento de arquitecturas:** De las arquitecturas implementadas en la biblioteca TSAI, se seleccionaron aquellas publicadas a partir del año 2020. Debido a las restricciones técnicas de

memoria del equipo de cómputo (8GB de VRAM), se omitieron los modelos basados en Transformers (TST, TSIT y ConvTran). Los nueve modelos restantes fueron entrenados con los datos de las series de tiempo, a los cuales se les aplicó previamente una normalización Z independiente por cada variable, paso crítico determinado en pruebas previas. La configuración de entrenamiento para todos los modelos constó de 100 épocas con un *early-stopping patience* de 10 épocas.

4.1.3. Resultados y discusión

Los resultados del entrenamiento y evaluación del primer experimento, contrastando los dos esquemas de validación cruzada, se resumen en la Tabla 4.3.

Tabla 4.3: Resultados del entrenamiento de modelos de clasificación con el *dataset* TSDS1

Modelo	<i>Stratified 5-Fold</i>	<i>StratifiedGroup 5-Fold</i>
HydraMultiRocket	0.8569 ± 0.0160	0.7144 ± 0.0829
Hydra	0.8893 ± 0.0120	0.6992 ± 0.0680
InceptionTime	0.9199 ± 0.0214	0.6699 ± 0.0494
MiniRocket	0.8616 ± 0.0107	0.6882 ± 0.0437
MultiRocket	0.8455 ± 0.0110	0.6962 ± 0.0650
OmniScaleCNN	0.9075 ± 0.0125	0.6760 ± 0.0935
TSPerceiver	0.8674 ± 0.0673	0.6229 ± 0.0355
TSSequencer	0.9039 ± 0.0178	0.6977 ± 0.0497
XCM	0.8504 ± 0.0150	0.6625 ± 0.0465

Los hallazgos demuestran el impacto negativo de la filtración de datos cuando no se agrupan correctamente las series de tiempo provenientes de una misma fuente. Al observar los resultados de la validación cruzada simple (*Stratified 5-Fold*), los modelos aparentan tener un desempeño muy alto; destacando InceptionTime con un 91.99 % de exactitud balanceada. Sin embargo, esta métrica es engañosa, ya que evalúa en gran medida la capacidad del modelo para memorizar patrones filtrados entre la prueba y el entrenamiento.

Al aplicar *StratifiedGroup 5-Fold*, el cual restringe severamente esta filtración aislando por video, el desempeño de todos los modelos disminuyó, reflejando su capacidad real de generalización. Bajo estas condiciones, HydraMultiRocket demostró ser la arquitectura más sólida con un *balanced accuracy* de 0.7144 ± 0.0829 , mientras que TSPerceiver reportó el menor desempeño con 0.6229 ± 0.0355 .

Finalmente, para complementar el análisis cuantitativo, la Figura 4.1 presenta la matriz de confusión obtenida por HydraMultiRocket en el Fold 4, la cual permite observar con mayor detalle la distribución de aciertos y errores entre las clases evaluadas.

A partir de esta matriz, se observa que los aciertos se concentran principalmente en la clase *normal*, donde se obtuvo una precisión de 92.94 % y una exhaustividad de 83.16 %. No obstante, debido al mayor volumen de segmentos temporales pertenecientes a esta categoría, los errores absolutos de clasificación desde *normal* hacia las clases minoritarias tienen un impacto importante sobre su precisión. En particular, parte de los segmentos normales fueron clasificados erróneamente como *evident* o *suspicious*, lo que incrementó los falsos positivos en dichas categorías.

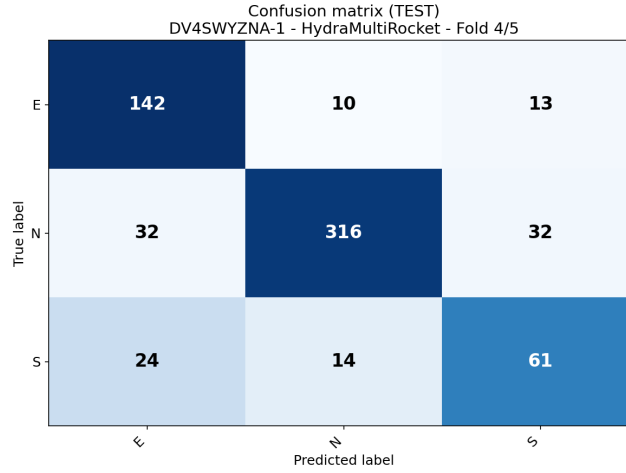


Figura 4.1: Matriz de confusión del *fold* 4 de 5 en el entrenamiento de HydraMultiRocket en el *dataset* TSDS1 con el esquema *StratifiedGroup 5-Fold Cross Validation*. *Balanced Accuracy* de 0.7695

Este efecto se aprecia especialmente en la clase *suspicious*, que fue la más difícil de reconocer para el modelo, con una precisión de 57.55 % y una exhaustividad de 61.62 %. En contraste, la clase *evident* alcanzó una precisión de 71.72 % y una exhaustividad de 86.06 %, mostrando una recuperación más alta de sus instancias reales, aunque también afectada por confusiones provenientes de la clase normal. Una posible causa de estas confusiones es la necesidad de detectar vehículos desde etapas muy tempranas de aparición en la escena, por ejemplo, cuando apenas comienzan a rebasar desde el borde de la imagen. Este aspecto puede mejorar al utilizar el modelo YOLO 11m entrenado en BDD100K, ya que permite una detección más temprana y estable de los vehículos involucrados en la interacción.

4.2. Experimento 2

4.2.1. Diseño experimental

- **Objetivo:** Evaluar el impacto de la técnica de aumento de datos mediante la permutación de *slots* en el rendimiento y la capacidad de generalización de los modelos de clasificación de series de tiempo, comparando los resultados contra la línea base establecida en el Experimento 1.
- **Tipo de diseño:** Diseño cuasiexperimental de corte transversal.
- **Objetos:** Las mismas nueve arquitecturas de clasificación de series de tiempo evaluadas en el experimento 1 (HydraMultiRocket, Hydra, InceptionTime, MiniRocket, MultiRocket, OmniScaleCNN, TSPerceiver, TSSequencer y XCM). En esta iteración, se utiliza el *dataset* generado con aumento de datos por permutaciones de *slots* (clave de control: TSDS2).
- **Instrumentos:** Equipo de cómputo descrito en la Tabla 4.1, y herramienta para el entrenamiento de modelos de clasificación de series de tiempo, descrita en la sección 3.7.2.
- **Proceso de evaluación y métricas:** Se mantiene el uso de la exactitud balanceada (*balanced accuracy*) en el conjunto de prueba como métrica principal. A diferencia del Experimento 1, para este conjunto de datos aumentado se utilizó únicamente el esquema de validación *StratifiedGroup*

Capítulo 4. Experimentación y Resultados

5-Fold Cross Validation. Esta decisión se tomó para asegurar de forma estricta que no existiera filtración de datos (*data leakage*) y evaluar la generalización.

4.2.2. Ejecución del experimento

- **Dataset:** Al conjunto base del Experimento 1 (TSDS1) se le aplicó el esquema de aumento de datos mediante permutaciones de *slots* con lo que se construyó el *dataset* TSDS2, que se utiliza en este experimento. Se configuró un mecanismo para seleccionar un máximo de 128 permutaciones por muestra, lo que equivale a solo un $\approx 0.038\%$ del espacio de permutaciones teórico (que puede alcanzar hasta 331,776 combinaciones posibles en series sin *slots* nulos). A pesar de esta selección reducida, el volumen de datos experimentó un incremento del $\approx 2697\%$, pasando de 2,818 a 78,814 segmentos de series de tiempo. La distribución resultante se detalla en la Tabla 4.4.

Tabla 4.4: Distribución de las clases y subconjuntos del *dataset* TSDS2 (con aumento de datos por intercambio de *slots*)

Clase	Conjunto de entrenamiento ($\approx 80\%$)	Conjunto de prueba ($\approx 20\%$)
<i>Evident</i>	13,997 (22.19 %)	3,499 (22.19 %)
<i>Normal</i>	40,802 (64.71 %)	10,201 (64.71 %)
<i>Suspicious</i>	8,252 (13.08 %)	2,063 (13.08 %)

- **Entrenamiento de arquitecturas:** Las nueve arquitecturas seleccionadas fueron entrenadas utilizando exactamente la misma configuración que en el Experimento 1: aplicación de normalización Z por variable en las series de tiempo, 100 épocas de entrenamiento y un *early-stopping patience* de 10 épocas.

4.2.3. Resultados y discusión

Los resultados del entrenamiento con el *dataset* aumentado se presentan en la Tabla 4.5.

Tabla 4.5: Resultados del entrenamiento de modelos de clasificación con el *dataset* TSDS2 (*Stratified-Group 5-Fold*)

Modelo	Exactitud Balanceada
HydraMultiRocket	0.7342 ± 0.0409
Hydra	0.6700 ± 0.0478
InceptionTime	0.7141 ± 0.0684
MiniRocket	0.7045 ± 0.0481
MultiRocket	0.7185 ± 0.0384
OmniScaleCNN	0.6898 ± 0.0847
TSPerceiver	0.7041 ± 0.0647
TSSequencer	0.6965 ± 0.0563
XCM	0.6933 ± 0.0562

Al comparar estos resultados con los obtenidos en el Experimento 1 bajo el mismo esquema de evaluación (*StratifiedGroup 5-Fold*), se confirma que la técnica de aumento de datos por intercambio de *slots* tiene un efecto positivo en la capacidad de generalización de las redes.

Las mejoras en el rendimiento (*balanced accuracy*) se observaron principalmente en aquellos modelos que originalmente presentaban el menor desempeño sin aumento de datos (por ejemplo, TSPerceiver pasó de 0.6229 a 0.7041). Además, la arquitectura HydraMultiRocket se mantuvo consistentemente

como el mejor modelo de todo el conjunto de pruebas, alcanzando un desempeño de 0.7342 ± 0.0409 . Esto corrobora empíricamente que someter los datos de entrenamiento a variaciones en la asignación espacial de los *slots* ayuda a la robustez del modelo, incluso cuando se utiliza una mínima fracción de las permutaciones teóricamente posibles.

4.3. Experimento 3

4.3.1. Diseño experimental

- **Objetivo:** Evaluar el desempeño de una selección de modelos en la clasificación de series de tiempo derivadas de videos de situaciones normales, sospechosas y evidentes de robo a transporte de carga en carretera utilizando un nuevo *dataset* que incorpora videos sintéticos y emplea un algoritmo de seguimiento más robusto ante dinámicas vehiculares no lineales.
- **Tipo de diseño:** Diseño cuasiexperimental de corte transversal.
- **Objetos:** Se evaluaron tres arquitecturas TSC de la familia Rocket: HydraMultiRocket, MiniRocket y MultiRocket. HydraMultiRocket se seleccionó por haber obtenido el mejor resultado en la línea base del Experimento 1; MiniRocket y MultiRocket se incluyeron para observar si las mejoras introducidas en el nuevo *dataset* y en el *pipeline* también repercutían en otras arquitecturas de la misma familia. El *dataset* evaluado corresponde a series de tiempo sin aumento por permutaciones (clave de control: TSDS3), por lo que el Experimento 1 constituye la referencia más directa para esta comparación.
- **Instrumentos:** Equipo de cómputo descrito en la Tabla 4.1, y herramienta para el entrenamiento de modelos de clasificación de series de tiempo, descrita en la sección 3.7.2.
- **Proceso de evaluación y métricas:** Se utilizó la exactitud balanceada (*balanced accuracy*) en el conjunto de pruebas como métrica principal para evaluar el rendimiento general mitigando el sesgo de las clases mayoritarias. A diferencia de los experimentos anteriores, el esquema de validación utilizado fue *Hold-Out* (*Fixed Train/Test splits*), conformando un único *split* manual. Este *split* se estructuró respetando una restricción estricta: excluir todos los videos sintéticos del conjunto de prueba para garantizar que la evaluación final se realizara únicamente sobre escenarios del mundo real.

4.3.2. Ejecución del experimento

- **Dataset:** El *dataset* TSDS3 se construyó a partir de una combinación de videos reales y sintéticos a 30 *fps* (90 normales, 86 sospechosos y 84 evidentes, aplicando exclusiones específicas en algunas muestras). Como mejora iterativa, DeepSORT se sustituyó por Deep OC-SORT en el módulo MOT, mientras que YOLO11s (COCO) se mantuvo en el módulo OD. La ventaja fundamental de Deep OC-SORT radica en que no asume dinámicas puramente lineales, lo que permite un seguimiento mucho más preciso frente a aceleraciones bruscas o cambios repentinos de trayectoria. Se extrajeron las características mediante un esquema de ventana deslizante de 5 segundos ($\ell_{\text{hom}} = \ell_{\text{req}} = 150$ observaciones), pero con un paso (*step*) reducido a 1 segundo para todas las clases, sin aplicar aumento de datos por permutaciones. Este proceso resultó en un total de 5,292 segmentos. La distribución de los subconjuntos de entrenamiento y prueba se detalla en la Tabla 4.6.

Tabla 4.6: Distribución de las clases y subconjuntos del *dataset* TSDS3 (*Hold-Out*)

Clase	Conjunto de entrenamiento ($\approx 80\%$)	Conjunto de prueba ($\approx 20\%$)
<i>Evident</i>	823 (19.71 %)	181 (16.20 %)
<i>Normal</i>	2840 (68.02 %)	765 (68.48 %)
<i>Suspicious</i>	512 (12.26 %)	171 (15.30 %)

- **Entrenamiento de arquitecturas:** Los tres modelos seleccionados se sometieron a una configuración de entrenamiento estandarizada de 100 épocas, utilizando un criterio de parada temprana (*early-stopping*) de 10 épocas, y aplicando normalización Z a las características de entrada. Adicionalmente, durante el entrenamiento se implementó la función de pérdida *weighted cross-entropy* para penalizar adecuadamente los errores en las clases minoritarias.

4.3.3. Resultados y discusión

Los resultados obtenidos en el conjunto de prueba para las tres arquitecturas evaluadas se muestran en la Tabla 4.7.

Tabla 4.7: Resultados del entrenamiento de modelos de clasificación con el *dataset* TSDS3 (*Hold-Out*)

Modelo	Exactitud Balanceada
HydraMultiRocket	0.8321
MultiRocket	0.8286
MiniRocket	0.8027

Los resultados del Experimento 3 fueron superiores a los de la línea base del Experimento 1, a pesar de no utilizar aumento de datos por permutaciones. HydraMultiRocket se posicionó nuevamente como la arquitectura con mayor exactitud balanceada, con un valor de 0.8321.

La mejora descriptiva se presentó en las tres arquitecturas de la familia Rocket. En el Experimento 1, con *StratifiedGroup 5-Fold*, HydraMultiRocket obtuvo 0.7144 ± 0.0829 , MiniRocket 0.6882 ± 0.0437 y MultiRocket 0.6962 ± 0.0650 . Los valores del Experimento 3 (0.8321, 0.8027 y 0.8286, respectivamente) superaron, en los tres casos, la media más una desviación estándar de la línea base. Sin embargo, esta comparación es únicamente descriptiva y no demuestra una diferencia estadísticamente significativa, ya que la desviación estándar entre *folds* no constituye un intervalo de confianza, el Experimento 3 utiliza una sola partición *hold-out* y ambos experimentos difieren en sus datos y configuración.

En conjunto, los resultados sugieren que los cambios incorporados en el Experimento 3 se asociaron con una mejora del rendimiento de la familia Rocket. No obstante, debido a que se modificaron simultáneamente la composición del *dataset*, la función de pérdida y el sistema MOT, el diseño experimental no permite atribuir la mejora de manera aislada a la incorporación de videos sintéticos, a la *weighted cross-entropy* o al uso de Deep OC-SORT.

El análisis de la matriz de confusión de HydraMultiRocket para este *dataset* (ver Figura 4.2) confirma una mejora general respecto al Experimento 1. En la clase *evident*, la precisión aumentó de 71.72 % a 88.33 %, mientras que la exhaustividad pasó de 86.06 % a 87.85 %, lo que indica una reducción importante de falsos positivos sin comprometer la recuperación de instancias reales. En la clase *normal*, también se observó un avance consistente, con una precisión que pasó de 92.94 % a 95.32 % y una exhaustividad que aumentó de 83.16 % a 90.46 %, reflejando una clasificación más estable de los segmentos normales.

En la clase *suspicious*, la precisión se mantuvo prácticamente estable, al pasar de 57.55 % a 57.82 %; sin embargo, la exhaustividad aumentó de 61.62 % a 71.35 %. Este comportamiento resulta relevante para el objetivo del sistema, ya que en escenarios de detección de comportamientos sospechosos es preferible recuperar una mayor proporción de casos potencialmente riesgosos, aun cuando esto implique aceptar una mayor cantidad de falsos positivos. En otras palabras, aunque todavía persiste cierta confusión entre las clases *normal* y *suspicious*, el modelo del presente experimento muestra una mayor capacidad para detectar segmentos sospechosos que podrían requerir atención posterior, lo cual robustece el sistema respecto a los experimentos iniciales.

Confusion matrix (TEST)
DV4SWYZNA-2 - HydraMultiRocket - Fold 1/1

	E	N	S
E	159	0	22
N	6	692	67
S	15	34	122
	E	N	S

True label

Predicted label

Figura 4.2: Matriz de confusión del entrenamiento de HydraMultiRocket en el *dataset* TSDS3. *Balanced Accuracy* de 0.8321

Capítulo 5

Conclusiones

En esta tesis se diseñó e implementó una arquitectura basada en visión artificial para la detección autónoma de comportamientos sospechosos asociados al robo a transporte de carga en carreteras, la cual alcanzó una exactitud balanceada del 83.21 % en la clasificación de eventos. La solución planteada adoptó como base la modularidad propuesta por Nazir et al., 2023 al integrar módulos de detección de objetos (OD), seguimiento múltiple de objetos (MOT) y clasificación de series de tiempo (TSC), pero extendió sus capacidades al abordar limitaciones identificadas en el estado del arte.

La arquitectura desarrollada en esta tesis destaca por las siguientes ventajas y aportaciones:

- **Mantenibilidad y flexibilidad del diseño:** La estructura modular permite actualizar o sustituir cada componente de forma independiente sin modificar el sistema completo. Esta característica facilita la incorporación de mejoras conforme evolucionan los modelos del estado del arte. A diferencia de los enfoques monolíticos, como las redes convolucionales 3D (3D-CNN) empleadas en Martínez et al., 2021, donde el procesamiento se realiza de extremo a extremo, la propuesta separa explícitamente las etapas de detección, seguimiento y clasificación. Cabe señalar que, en dicho trabajo, el desempeño disminuye de 85.71 % en un escenario balanceado a 78.26 % en uno no balanceado; este último resulta más comparable con las condiciones de esta investigación.

Más allá de la comparación de resultados, la modularidad permite facilitar la división del trabajo para potencialmente mejorar en el futuro este desarrollo de forma ágil, permitiendo que distintas personas o equipos trabajen de manera independiente sobre cada módulo; por ejemplo, uno enfocado en robustecer la detección de actores en carretera en escenarios adversos desde una perspectiva de primera persona, y otro en el desarrollo o ajuste de modelos de clasificación de series de tiempo. Asimismo, el diseño permite incorporar de manera directa otros modelos que surjan en el estado del arte de forma externa. Por ejemplo, si otros investigadores diseñan e implementan en una biblioteca un modelo de clasificación de series de tiempo más eficiente o preciso que los actuales, este puede integrarse sustituyendo únicamente dicho módulo con un impacto directo en el desempeño general, sin necesidad de rediseñar el sistema completo. Dado que la arquitectura opera sobre representaciones cinemáticas previamente generadas, los modelos de clasificación pueden entrenarse sin reprocesar los videos originales, lo que reduce potencialmente los tiempos de experimentación.

- **Interpretabilidad cinemática e integridad temporal:** La arquitectura desarrollada utiliza características cinemáticas simples, como la posición (x, y) y las dimensiones (w, h) de los *bounding boxes*, cuyo comportamiento refleja la interacción de los agentes en la escena (automóviles,

camiones, motocicletas y personas). A partir de estas variables se generan representaciones en forma de series de tiempo, las cuales pueden visualizarse mediante diagramas que permiten un análisis detallado y transparente.

El enfoque de Nazir et al., 2023 también construye series de tiempo a partir de variables cinemáticas similares. Sin embargo, presenta una limitación estructural relevante: las trayectorias de múltiples individuos son aplanadas en una única secuencia, lo que rompe la continuidad temporal individual. Además, en ningún momento se valida de forma cualitativa que las series de tiempo representen correctamente las escenas trabajadas.

En contraste, la arquitectura desarrollada en esta tesis preserva la identidad y la trayectoria de cada agente de forma independiente. Esto permite mantener la continuidad temporal de cada actor y analizar sus interacciones de manera individual, facilitando una interpretación coherente de la dinámica de la escena.

- **Contribución de nuevos conjuntos de datos:** La revisión de la literatura reveló una brecha evidente: los trabajos previos abordan comportamientos delictivos estrictamente en entornos de circuito cerrado (CCTV) con cámaras fijas, o bien, analizan dinámicas vehiculares complejas pero limitadas a accidentes viales mediante cámaras móviles (*dashcams*). Hasta donde se identificó en la revisión de la literatura, esta investigación constituye el primer trabajo que aporta un conjunto de datos diseñado específicamente para el problema de robos a transporte de carga vistos desde la perspectiva de primera persona, el cual integra secuencias reales obtenidas de redes sociales y sintéticas generadas mediante GTA V, American Truck Simulator y Veo 3 Fast.

- **Rigor metodológico y exhaustividad en la evaluación:** La validación cuantitativa de los modelos de clasificación se realizó considerando el desbalance de clases presente en el conjunto de datos, utilizando como métrica principal el *balanced accuracy*. Asimismo, se evitó la filtración de datos asociada al uso de ventanas deslizantes, garantizando que las series de tiempo de un mismo origen no se distribuyeran entre los conjuntos de entrenamiento y prueba. Con ello, se aseguró que la evaluación reflejara la capacidad de generalización del modelo de clasificación de series de tiempo, y no su memorización.

En contraste con metodologías evaluadas bajo enfoques acotados —como la de Nazir et al. (2023), que comparó únicamente 4 modelos de clasificación de series de tiempo, donde MiniRocket obtuvo un *F1-score* medio de 0.89 ± 0.09 —, en esta tesis se realizó un *benchmark* más amplio, evaluando 9 modelos distintos. Los modelos finales se probaron exclusivamente con videos reales.

Adicionalmente, en el capítulo de libro de Gutiérrez Montor et al. (2027), derivado de esta tesis, se describe la metodología empleada para comparar el tiempo de inferencia de los modelos de clasificación en una etapa *offline*, utilizando el conjunto de datos completo. Esto contrasta con la evaluación basada en un único video de 14 segundos a 30 fps reportada en Nazir et al. (2023), y proporciona métricas más representativas para configurar los parámetros operativos del sistema en tiempo real.

- **Viabilidad operativa mediante el *pipeline* asíncrono:** La arquitectura implementa un *pipeline* asíncrono que mitiga las restricciones computacionales asociadas al procesamiento continuo de video. Es decir, no se restringe a procesar solo videos ya almacenados como archivos. Este diseño desacopla la adquisición de fotogramas de la inferencia de los modelos mediante colas de procesamiento intermedias, lo que evita bloqueos en el hilo de captura y la acumulación de latencia. Para ello, se omiten o descartan fotogramas cuando la capacidad de procesamiento no permite analizarlos oportunamente, conservando siempre la observación más reciente disponible.

La validez de este esquema se verificó de manera cualitativa en secuencias de video largas ingresadas al sistema de forma continua mediante OBS Studio, mostrando la viabilidad del enfoque para operar en tiempo real bajo condiciones controladas. Esto establece una base técnica para

su futura implementación en dispositivos de procesamiento en el borde (*edge computing*) dentro de escenarios operativos reales.

Tabla 5.1: Evidencias del cumplimiento de los objetivos específicos

Objetivo Específico	Evidencia del cumplimiento
1. Comparar experimentalmente al menos tres modelos de detección de objetos, dos modelos de seguimiento de objetos y tres modelos de clasificación de series de tiempo, que de acuerdo con el estado del arte resulten más adecuados para resolver el problema.	Aunque se hayan probado menos modelos OD, se evaluó una mayor cantidad de sistemas MOT y modelos TSC que los planteados originalmente: ■ Dos modelos de detección de objetos: YOLO11s entrenado en el <i>dataset</i> COCO, y YOLO11m entrenado en el <i>dataset</i> BDD100K. ■ Tres sistemas MOT: DeepSORT, ByteTrack y Deep OC-SORT. ■ Nueve modelos de clasificación de series de tiempo: HydraMultiRocket, Hydra, InceptionTime, MiniRocket, MultiRocket, OmniScaleCNN, TSPerceiver, TSSequencer y XCM. Los resultados se reportan en el Capítulo 4 de este documento y pueden consultarse en extenso en el enlace siguiente: https://drive.google.com/drive/folders/10-NiUTZFHDz7ZzrKTQsCR4kvNu0CwriO?usp=sharing
2. Crear un <i>dataset</i> de videos sintéticos, conformado por videos de situaciones sospechosas típicas del robo a transporte de carga y videos de situaciones normales.	Se generó un <i>dataset</i> de videos sintéticos, con 15 videos de situaciones normales, 10 sospechosas y 7 evidentes, disponible en: https://drive.google.com/drive/folders/1uBc-2mu8EHZjbcXaGdiWBXvAvWnVKTJJ?usp=drive_link
3. Crear un <i>dataset</i> de videos reales, conformado por videos de situaciones sospechosas típicas del robo a transporte de carga y videos de situaciones normales.	Se generó un <i>dataset</i> de videos reales, con 84 videos de situaciones normales, 82 sospechosas y 79 evidentes, disponible en: https://drive.google.com/drive/folders/1PIJgFpZr49C14z91rNMIFnBiQ8PwW0IU?usp=drive_link
4. Proponer una combinación de técnicas de visión artificial y una arquitectura de aprendizaje profundo para detectar comportamientos sospechosos característicos del robo a transporte de carga.	En el Capítulo 3 se presentan la propuesta de solución y su implementación por computadora. La materialización de esta implementación puede consultarse en: https://drive.google.com/drive/folders/1Ge-jHLnfmmfscxLsK5N__d-yV6HTVFyP?usp=drive_link

Capítulo 6

Trabajo futuro

Para incrementar el *balanced accuracy* del modelo y optimizar su eficiencia computacional, se plantean las siguientes líneas de investigación y desarrollo como posibilidades a explorar:

1. Optimización en los modelos de clasificación de series de tiempo

Se propone investigar la reducción del tamaño de los modelos y el uso más eficiente de la memoria disminuyendo los *slots* vacíos en las series de tiempo. Una estrategia viable consistiría en colapsar clases con cinemáticas similares en superclases (por ejemplo, agrupar *truck*, *car* y *bus* en la superclase *vehicle*). Esto permitiría asignar *slots* de manera dinámica (ej. 6 para vehículos, 5 para personas), reduciendo la dispersión espacial del modelo. Además, esta agrupación podría eliminar el *flickering* (fluctuación) entre clases visualmente similares, lo que haría prescindible el mecanismo actual de estabilización de clases.

2. Robustecimiento de la detección y seguimiento de objetos (OD y MOT)

Dado que el desempeño general del sistema depende fuertemente de la detección inicial, se sugieren las siguientes mejoras:

- a. **Entrenamiento de un nuevo modelo de detección:** Actualmente, YOLO11s (COCO) presenta falsos positivos (como detectar el cofre del ego-vehículo como otro auto) y falsos negativos (vehículos con puertas abiertas). Por su parte, YOLO11m (BDD100K) mitiga algunos errores y permite detectar mejor los vehículos en ambientes nocturnos y en videos de baja resolución, pero falla en la detección de personas y clasifica erróneamente las puertas abiertas. Por lo tanto, resulta conveniente un entrenamiento específico con imágenes de entornos de carretera grabados desde cámaras de tablero (*dashcams*), considerando condiciones adversas (baja iluminación, clima, baja resolución, etc.).
- b. **Transición a arquitecturas JDE (*Joint Detection and Embedding*):** Para mejorar el tiempo de inferencia, se podría explorar la unificación de la detección (OD) y la reidentificación (parte del subsistema MOT) en un solo modelo, eliminando la carga de la asociación visual por separado. La arquitectura diseñada debería tener mecanismos de seguimiento robustos ante trayectorias no lineales y aceleraciones bruscas, similares a Deep OC-SORT.

3. Integración de características del entorno

Se plantea la posibilidad de incorporar variables que caractericen el comportamiento de la carretera sin perder la simplicidad computacional. Las opciones incluyen la estimación de un punto de fuga

dinámico basado en la geometría del camino o en el movimiento del flujo vehicular, así como el uso de flujo óptico sobre el asfalto para determinar la dirección del ego-vehículo.

4. Validación y *benchmarking* extendido

- a. **Migración de datos:** Se contempla el uso de la versión 6 del *dataset* de videos reales y sintéticos, la cual obedece a una arquitectura de almacenamiento y organización más estricta¹⁸. Este *dataset* busca reducir la posibilidad de *garbage-in garbage-out* (GIGO) en las clases *suspicious* y *evident*, aumentar la diversidad y reducir *data-leakage* implícito en la clase *normal* sin desbalancear más el *dataset*.
- b. **Comparativa con el estado del arte:** Sería pertinente realizar experimentos formales para comparar la arquitectura diseñada contra enfoques tradicionales (por ejemplo, CNN 3D, CNN-LSTM o Vision Transformers). El objetivo de estos experimentos sería comprobar empíricamente ventajas teóricas de la propuesta —que de hecho no se han mencionado explícitamente— tales como: el requerimiento de una menor cantidad de datos de entrenamiento, la capacidad de analizar ventanas temporales más largas y la viabilidad de ejecución en tiempo real en dispositivos *edge*.

¹⁸Esta versión del *dataset*, la cual no fue utilizada en los experimentos reportados, así como sus especificaciones, puede consultarse en: https://drive.google.com/drive/folders/1mY4lYXhvbqH4u3eIva0fLyCEDD753dEZ?usp=drive_link

Anexos

Anexo A

Especificación del *dataset* de videos reales

En la Figura A.1 se muestran los fotogramas de dos videos representativos de cada una de las clases consideradas en el *dataset* de videos reales.



Figura A.1: Fotogramas representativos de dos videos de cada una de las clases (*normal*, *suspicious*, *evident*) del *dataset* de videos reales.

Anexo A. Especificación del *dataset* de videos reales

Estas clases fueron definidas tomando como referencia las categorías “*Crime Evidence Segment (CES)*” y “*Suspicious Behavior Segment (SBS)*” propuestas por Martínez et al. (2021), pero adaptándolas al fenómeno del robo a transporte de carga. A continuación se describe cada una de ellas:

- **Normal (normal):** segmentos de video donde se observa un comportamiento vehicular esperado en una carretera. Por ejemplo, vehículos rebasando en el carril izquierdo, vehículos circulando en el sentido contrario en su respectivo carril, o desplazamiento a baja velocidad con un vehículo próximo al frente, entre otras situaciones.
- **Suspicious (sospechoso):** segmentos de video en los que un vehículo realiza maniobras inusuales como rebasar para inmediatamente bloquear el paso mediante zigzagueos o frenadas bruscas e irregulares, donde todavía no es posible confirmar que se trate de un caso de robo a transporte de carga, pero ya supone una situación de riesgo.
- **Evident (evidente):** segmentos de video donde se observan personas armadas agrediendo directamente al conductor de la unidad. Generalmente ocurre cuando la unidad asaltada y el vehículo que va delante quedan completamente detenidos, y los agresores descienden del vehículo frontal para acercarse al conductor.

La versión 4 del *dataset* tiene 245 videos, distribuidos de acuerdo con lo que se muestra en la Figura A.2¹⁹.

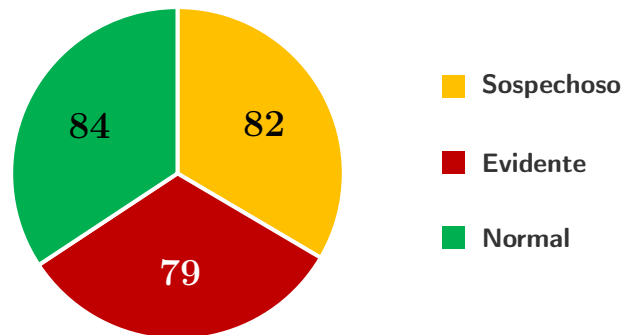


Figura A.2: Distribución de las clases del *dataset* de videos reales (versión 4) con 245 videos.

A los videos se les hizo el siguiente tratamiento:

- Se hizo un *trimming* para extraer los segmentos relevantes de cada una de las clases consideradas en el *dataset*.
- Se hizo un *cropping* y *scaling* para normalizar todos los videos a una resolución 1080p en formato 16:9. La diferencia, principalmente en el formato de forma, puede afectar severamente a la generación de las series de tiempo.
- Se hizo un *ROI masking* en una pequeña cantidad de videos, para eliminar marcas de agua, transiciones, subtítulos, entre otros elementos.

¹⁹El *dataset* de videos reales se encuentra disponible en: https://drive.google.com/drive/folders/1QvDZvXMSyTbU_SSb94GfHZbyRlrOhHtT?usp=drive_link

Anexo A. Especificación del *dataset* de videos reales

- ***Time scaling***: En algunos videos se normalizó la velocidad, ya que, principalmente en videos que compilan varias situaciones, suelen tener aceleradas ciertas partes del video para que el consumidor de contenido en redes sociales se centre en las partes importantes, y evitar que el video dure una gran cantidad de tiempo.

Todos los videos tienen una resolución 1080p, en formato 16:9, y 29.97 (NTSC) o 30 *fps*.

Anexo B

Especificación del *dataset* de videos sintéticos

Para la generación de videos sintéticos se utilizaron 3 herramientas: los videojuegos GTA V²⁰ y American Truck Simulator²¹, y el generador de videos Veo 3 Fast²².

La selección de estas tres herramientas fue el resultado de una evaluación cualitativa que abarcó diversas alternativas, incluyendo modelos de IA generativa de video (OpenAI, Runway, Alibaba Qwen, Nanonoble, StreamingT2V y DreamForge), modelos de mundo para conducción autónoma (GAIA-2 y DriveArena) y simuladores vehiculares avanzados (CARLA y BeamNG). En dicha evaluación se contrastaron criterios determinantes: los permisos legales para uso de contenido con fines académicos y no comerciales, el realismo en las cinemáticas, la duración máxima de los videos generados, el grado de consistencia, la controlabilidad de los escenarios y agentes, los requerimientos de *hardware* y los costos asociados²³.

Se siguieron los siguientes criterios de exclusión en el *dataset* de los videos generados sintéticamente:

- El video no simula una escena grabada desde una *dashcam*.
- El robo o asalto ocurre como observador y no como víctima.
- Los objetos o personas entran en escena o siguen trayectorias de manera inconsistente.
- Los movimientos y aceleraciones de los objetos y personas son irreales e inestables.

Las tres herramientas elegidas son útiles para generar videos de la clase *normal*. Sin embargo, solo Veo 3 Fast es útil para generar videos de las clases *sospechoso* y *evidente* ya que, por un lado, American Truck Simulator no es capaz de simular personas como sí es posible en GTA V, mientras que en GTA V, aunque sí es posible simular personas, el movimiento y aceleración de los vehículos y agentes no es muy realista ni estable.

²⁰La licencia de uso del contenido generado con este videojuego puede consultarse en la página de Rockstar Games: <https://www.rockstargames.com/mx/legal>.

²¹La licencia de uso y distribución de contenido puede consultarse en la página de SCS Software: https://www.scssoft.com/video_sharing_licence.pdf.

²²Las condiciones del servicio y políticas de uso pueden consultarse en la página de Google: <https://policies.google.com/terms>.

²³El análisis detallado y la evaluación comparativa de estas herramientas pueden consultarse en: <https://drive.google.com/drive/folders/13LYQVtMS0K1xNU9ADi8wQ3RHUOxEjhXu?usp=sharing>

Los videos recibieron el siguiente tratamiento:

- **Time scaling:** normalización de velocidad, ya que en algunos casos se pidió en el *prompt* que se simulara un video en cámara rápida para poder incorporar una mayor duración del suceso en el tiempo acotado de 8 segundos posibles de generar en Veo 3 Fast.
- **Trimming:** de cada video generado se pueden extraer uno o más segmentos de las clases *normal*, *suspicious* y *evident*.
- **Scaling:** normalización a la resolución 1920 x 1080p.
- A diferencia de los videos reales, el *cropping* y *masking* fueron innecesarios, al generarse videos sin marcas de agua y todos con formato 16:9.

En total, actualmente en el *dataset* de videos sintéticos se cuenta con 32 videos, distribuidos de acuerdo con lo que se presenta en el gráfico de la Figura B.1 ²⁴.

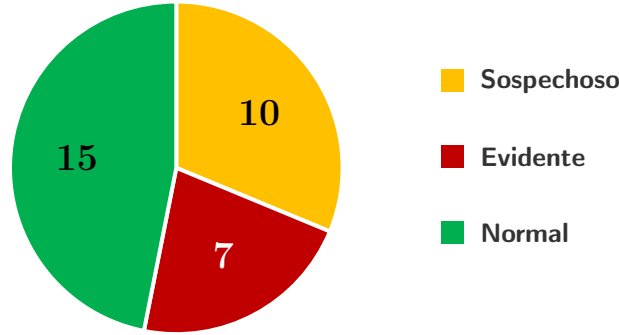


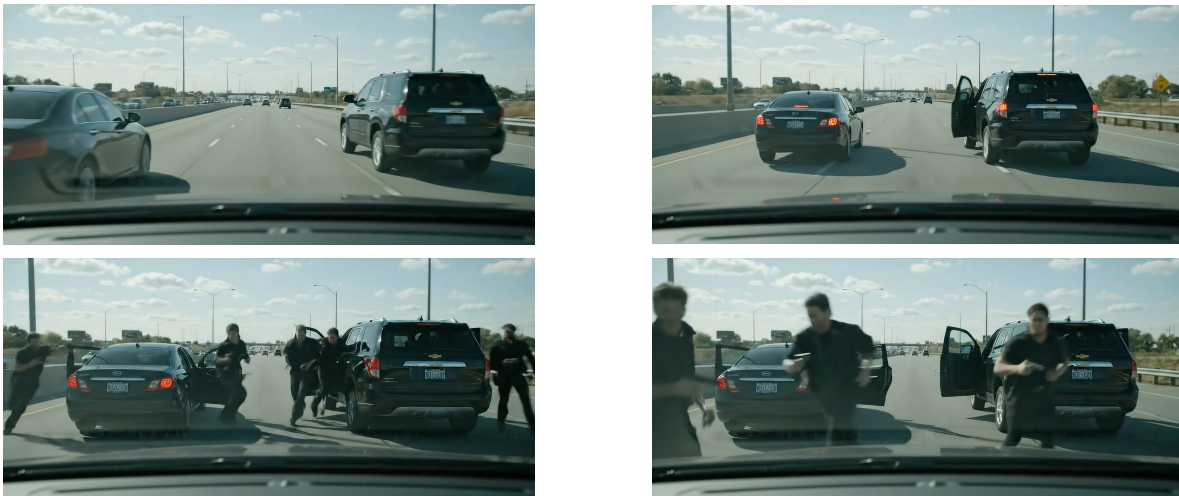
Figura B.1: Distribución de las clases del *dataset* de videos sintéticos (versión 4) con 32 videos.

La tasa de éxito de los videos generados en Veo 3 Fast del 9 de julio al 22 de septiembre de 2025 es del 12.34 % (de 81 videos generados 10 son funcionales, de los cuales se extrajeron 10 segmentos de la clase *sospechoso* y 7 segmentos de la clase *evidente*). En la Figura B.2 se presenta un ejemplo de los videos generados con Veo 3 Fast de las clases *sospechoso/evidente* (todavía sin segmentación o *trimming*) que cumplió con los criterios para ser incluido.

El principal criterio de exclusión por el cual se determinó que un video generado por Veo 3 Fast no era un caso exitoso es que el robo o asalto ocurría como espectador y no como víctima, es decir, que a pesar de que se simulaba la grabación desde una *dashcam*, el asalto le ocurría a un tercer vehículo presente al frente, y en muchas ocasiones dicho vehículo aparecía de una forma inconsistente. En la Figura B.3 se presentan capturas de uno de estos casos y su respectivo *prompt*.

²⁴El *dataset* de videos sintéticos se encuentra disponible en: https://drive.google.com/drive/folders/1d9vzQsUrzCmhsTN_8-nQCGGlp90IN8Cq?usp=drive_link

Anexo B. Especificación del *dataset* de videos sintéticos



Prompt:

"View from a front-facing dashcam while driving on the highway. As I'm driving, a car overtakes me on the left, followed by another vehicle that also speeds ahead. Both cars then brake in front of me, forcing me to stop (all vehicles come to a halt). Two armed individuals get out of one car and three from the other car, and run toward my vehicle with the clear intention of assaulting me. Everything unfolds with realistic movements, speeds, and accelerations"

Figura B.2: *Prompt* utilizado y fotogramas clave del video "Ve0 E4.mp4" del *dataset* de videos sintéticos.



Prompt:

"This video is filmed from a fixed dashcam mounted inside my truck, showing a highway scene at medium-low driving speed. A car overtakes my truck, the same car abruptly cuts in front and stops, forcing me to brake and come to a complete stop. Moments later, three offenders exit the same car one by one and walk toward my truck. As they approach the dashcam view, they spread to the sides of my vehicle while aiming guns at me. The video should play at 1.5x speed, but all actions must remain consistent with realistic relative timing and duration."

Figura B.3: *Prompt* utilizado y fotogramas clave de un video generado en Veo 3 Fast no incluido en el *dataset* de videos sintéticos.

Referencias

- Abdullah, F., & Jalal, A. (2023). Semantic Segmentation Based Crowd Tracking and Anomaly Detection via Neuro-fuzzy Classifier in Smart Surveillance System. *Arabian Journal for Science and Engineering*, 48, 2173-2190. <https://doi.org/10.1007/s13369-022-07092-x>
- Ahmed, A. A., & Echi, M. (2021). Hawk-Eye: An AI-Powered Threat Detector for Intelligent Surveillance Cameras. *IEEE Access*, 9, 63283-63293. <https://doi.org/10.1109/ACCESS.2021.3074319>
- Ali, M. M. (2023). Real-time video anomaly detection for smart surveillance. *IET Image Processing*, 17, 1375-1388. <https://doi.org/10.1049/ipr2.12720>
- Alvarado, N. F. (2025). Torres de inspección vehicular de la Guardia Nacional; una herramienta contra el robo de autos. *La Prensa*. Consultado el 5 de marzo de 2026, desde <https://oem.com.mx/la-prensa/policiaca/guardia-nacional-activa-torres-de-inspeccion-en-carreteras-para-frenar-el-robo-de-vehiculos-26253200>
- ANERP.V. (2026). *Sitio Oficial - Innovación y Seguridad: Plataforma Centinela 2.0*. Consultado el 5 de marzo de 2026, desde <https://www.anerpv.mx/>
- Awwad, Y. A., Rana, O., & Perera, C. (2024). Anomaly Detection on the Edge Using Smart Cameras under Low-Light Conditions. *Sensors*, 24. <https://doi.org/10.3390/s24030772>
- CANACAR. (2025). *Agenda Económica del Autotransporte de Carga 2025* (inf. téc.). Cámara Nacional del Autotransporte de Carga. <https://canacar.com.mx/servicios/estadistica/>
- Cao, W., Wang, D., Li, J., Zhou, H., Li, L., & Li, Y. (2018). BRITS: Bidirectional Recurrent Imputation for Time Series. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 6775-6785.
- Chang, C. W., Chang, C. Y., & Lin, Y. Y. (2022). A hybrid CNN and LSTM-based deep learning model for abnormal behavior detection. *Multimedia Tools and Applications*, 81, 11825-11843. <https://doi.org/10.1007/s11042-021-11887-9>
- Chen, S., Sun, W., Huang, L., Li, X., Wang, Q., & John, D. (2024). POCKET: Pruning random convolution kernels for time series classification from a feature selection perspective. *Knowledge-Based Systems*, 300, 112253. <https://doi.org/10.1016/j.knosys.2024.112253>
- Chiuan, W. K., Ridzuan, M., & Ahmad, B. (2024). Reviewing Approaches and Techniques for Detecting Suspicious Human Behavior: A Comprehensive Survey. *ELEKTRIKA, Journal of Electrical Engineering*, 23, 44-52.
- Cortés Ramírez, F. (2022, diciembre). *Reporte de resultados de la tesis de Maestría en Ciencias: Detección automática de casos de robo a transportistas usando redes neuronales* (Reporte de resultados). Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). Cuernavaca, Morelos, México.
- Dai, M., Yuan, J., Liu, H., & Wang, J. (2024). TSCF: An Improved Deep Forest Model for Time Series Classification. *Neural Processing Letters*, 56(1), 13. <https://doi.org/10.1007/s11063-024-11531-1>
- Dempster, A., Petitjean, F., & Webb, G. I. (2020). ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5), 1454-1495. <https://doi.org/10.1007/s10618-020-00701-z>
- Dempster, A., Schmidt, D. F., & Webb, G. I. (2021). MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 248-257. <https://doi.org/10.1145/3447548.3467231>
- Dempster, A., Schmidt, D. F., & Webb, G. I. (2023). HYDRA: Competing convolutional kernels for fast and accurate time series classification. *Data Mining and Knowledge Discovery*, 37(5), 1779-1805. <https://doi.org/10.1007/s10618-023-00939-3>

Referencias

- Dempster, A., Schmidt, D. F., & Webb, G. I. (2024). quant: a minimalist interval method for time series classification. *Data Mining and Knowledge Discovery*, 38(4), 2377-2402. <https://doi.org/10.1007/s10618-024-01036-9>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=YicbFdNTTy>
- Du, W. (2023). PyPOTS: a Python toolbox for data mining on Partially-Observed Time Series. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2305.18811>
- Du, W., Côté, D., & Liu, Y. (2023). SAITS: Self-attention-based imputation for time series. *Expert Systems with Applications*, 219, 119619. <https://doi.org/10.1016/j.eswa.2023.119619>
- Erregue, I., Nasrollahi, K., & Escalera, S. (2025). YOLO11-JDE: Fast and Accurate Multi-Object Tracking with Self-Supervised Re-ID. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. <https://doi.org/10.1109/WACVW65960.2025.00092>
- Fang, J., Qiao, J., Xue, J., & Li, Z. (2024). Vision-Based Traffic Accident Detection and Anticipation: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 34, 1983-1999. <https://doi.org/10.1109/TCSVT.2023.3307655>
- Faouzi, J. (2022). Time Series Classification: A review of Algorithms and Implementations. En K. Kotecha (Ed.), *Machine Learning: Emerging Trends and Applications*. Proud Pen. <https://hal.inria.fr/hal-03558165>
- Faouzi, J., & Janati, H. (2020). pyts: A Python Package for Time Series Classification. *Journal of Machine Learning Research*, 21(46), 1-6. <http://jmlr.org/papers/v21/19-763.html>
- Fauvel, K., Lin, T., Masson, V., Fromont, É., & Termier, A. (2021). XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification. *Mathematics*, 9(23), 3137. <https://doi.org/10.3390/math9233137>
- Foumani, N. M., Tan, C. W., Webb, G. I., & Salehi, M. (2024). Improving position encoding of transformers for multivariate time series classification. *Data Mining and Knowledge Discovery*, 38(1), 22-48. <https://doi.org/10.1007/s10618-023-00948-2>
- Garlem, J. (2025). Torres de inspección, tecnología de la GN al servicio de la seguridad. *La Crónica de Hoy*. Consultado el 5 de marzo de 2026, desde <https://www.cronica.com.mx/nacional/2025/08/13/torres-de-inspeccion-tecnologia-de-la-gn-al-servicio-de-la-seguridad/>
- Geotab. (2026). *Software de Gestión de Flotas de Transporte*. Consultado el 5 de marzo de 2026, desde <https://www.geotab.com/es-latam/software-gestion-de-flotas/>
- Gómez, A. (2023). AI27 presenta AiS, plataforma basada en IA para prevenir el robo al transporte. *Revista TyT*. Consultado el 5 de marzo de 2026, desde <https://www.tyt.com.mx/nota/ai27-presenta-ais-plataforma-basada-en-ia-para-prevenir-el-robo-al-transporte>
- Guardia Nacional. (2021). *GN Carreteras: Servicio de alertas digitales*. Gobierno de México. Consultado el 5 de marzo de 2026, desde <https://www.gob.mx/guardianacional/articulos/servicio-de-alertas-digitales-gn-carreteras>
- Gutiérrez Montor, M., González Franco, N., Fuentes Pacheco, J., González Serna, J. G., & Castro Sánchez, N. A. (2027). Benchmarking Recent Architectures for Multivariate Time Series Classification: Effectiveness vs. Efficiency [En prensa]. En C. Gunupudi & R. K. Gnanasekaran (Eds.), *AI, Edge and Blockchain Systems* (Vol. 11). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-36532-3_10
- IBM Corporation. (2025). *Irregular time series* [Accessed: 2025-11-19]. IBM Informix V15.0.0 documentation. <https://www.ibm.com/docs/en/informix-servers/15.0.x?topic=concepts-irregular-time-series>
- Indicador Automotriz. (2025). ANERPV presentó Centinela 2.0, la evolución de la seguridad logística en México. *Indicador Automotriz*. Consultado el 5 de marzo de 2026, desde <https://www.indicadorautomotriz.com.mx/tendencias/anerpv-presento-centinela-2-0-la-evolucion-de-la-seguridad-logistica-en-mexico/>
- INEGI. (2024). *Encuesta Nacional de Victimización de Empresas (ENVE) 2024, principales resultados* (inf. téc.). Instituto Nacional de Estadística y Geografía. https://www.inegi.org.mx/contenidos/programas/enve/2024/doc/enve_2024_presentacion_ejecutiva.pdf

- Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4), 917-963. <https://doi.org/10.1007/s10618-019-00619-1>
- Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., & Petitjean, F. (2020). InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6), 1936-1962. <https://doi.org/10.1007/s10618-020-00710-y>
- Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., & Forestier, G. (2023). LITE: Light Inception with boosting techniques for Time Series Classification. *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, 1-10. <https://doi.org/10.1109/DSAA60987.2023.10302569>
- Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., Hénaff, O., Botvinick, M. M., Zisserman, A., Vinyals, O., & Carreira, J. (2022). Perceiver IO: A General Architecture for Structured Inputs & Outputs. *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=fILj7WpI-g>
- Karim, M. M., Yin, Z., & Qin, R. (2024). An Attention-Guided Multistream Feature Fusion Network for Early Localization of Risky Traffic Agents in Driving Videos. *IEEE Transactions on Intelligent Vehicles*, 9, 1792-1803. <https://doi.org/10.1109/TIV.2023.3275543>
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision – ECCV 2014*, 740-755. https://doi.org/10.1007/978-3-319-10602-1_48
- Lin, W., Gao, J., Wang, Q., & Li, X. (2021). Learning to detect anomaly events in crowd scenes from synthetic data. *Neurocomputing*, 436, 248-259. <https://doi.org/10.1016/j.neucom.2021.01.031>
- Lis, K., Honari, S., Fua, P., & Salzmann, M. (2024). Detecting Road Obstacles by Erasing Them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46, 2450-2460. <https://doi.org/10.1109/TPAMI.2023.3335152>
- Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., & Long, M. (2024). iTransformer: Inverted Transformers are Effective for Time Series Forecasting. *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=JePfAI8fah>
- Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. J. (2019). sktime: A unified interface for machine learning with time series. *NeurIPS Workshop on Systems for ML*.
- Maggiolino, G., Ahmad, A., Cao, J., & Kitani, K. (2023). Deep OC-Sort: Multi-Pedestrian Tracking by Adaptive Re-Identification. *Proceedings - International Conference on Image Processing, ICIP*, 3025-3029. <https://doi.org/10.1109/ICIP49359.2023.10222576>
- Martínez, G. A., Abreu, J. R., Ortiz, J. C., Garcia, A., & Terashima, H. (2021). Criminal intention detection at early stages of shoplifting cases by using 3D convolutional neural networks. *Computation*, 9(2), 25. <https://doi.org/10.3390/computation9020024>
- Mehmood, A. (2021). LightanomalyNet: A lightweight framework for efficient abnormal behavior detection. *Sensors*, 21, 8501. <https://doi.org/10.3390/s21248501>
- Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Guijo-Rubio, D., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., & Bagnall, A. (2024). aeon: a Python Toolkit for Learning from Time Series. *Journal of Machine Learning Research*, 25(289), 1-10. <http://jmlr.org/papers/v25/23-1444.html>
- Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., & Bagnall, A. (2021). HIVE-COTE 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(11-12), 3211-3243. <https://doi.org/10.1007/s10994-021-06057-9>
- Mumtaz, A., Sargano, A. B., & Habib, Z. (2023). Robust learning for real-world anomalies in surveillance videos. *Multimedia Tools and Applications*, 82, 20303-20322. <https://doi.org/10.1007/s11042-023-14425-x>
- Navixy. (2026). *Advertencias ADAS - User guide*. Consultado el 5 de marzo de 2026, desde <https://navixy.com/docs/user/es/guide/reports/specific-report-details/report-on-all-events>
- Nazir, A., Mitra, R., Sulieman, H., & Kamalov, F. (2023). Suspicious Behavior Detection with Temporal Feature Extraction and Time-Series Classification for Shoplifting Crime Prevention. *Sensors*, 23(13), 5811. <https://doi.org/10.3390/s23135811>
- Oguiza, I. (2023). tsai: A state-of-the-art deep learning library for time series and sequential data. <https://github.com/timeseriesAI/tsai>

Referencias

- Opitz, J. (2024). A Closer Look at Classification Evaluation Metrics and a Critical Reflection of Common Evaluation Practice. *Transactions of the Association for Computational Linguistics*, 12, 820-836. https://doi.org/10.1162/tacl_a_00675
- ORUSS & CANACAR. (2025). *Estrategia BALAM* (Reporte sectorial). Consultado el 5 de marzo de 2026, desde <https://blog.oruss.io/content/files/2025/02/Balam.pdf>
- Overhaul. (2026a). *Gestión de Riesgos en la Cadena de Suministro y Visibilidad en Tiempo Real*. Consultado el 5 de marzo de 2026, desde <https://www.over-haul.com/es/>
- Overhaul. (2026b). *México: Reporte Anual 2025 de robo a transporte de carga* (inf. téc.). <https://www.over-haul.com/es/reports/mexico-annual-cargo-theft-report-2025>
- Qian, L., Raj, J. A., Ellis, H. L., Zhang, A., Zhang, Y., Wang, T., Dobson, R. J., & Ibrahim, Z. (2023). CSAI: Conditional Self-Attention Imputation for Healthcare Time-series. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2312.16713>
- Samsara. (2026). *Gestión de flotas y telemática: Rastreo GPS, seguridad en video y monitoreo de sensores*. Consultado el 5 de marzo de 2026, desde <https://www.samsara.com/mx/fleet-management/>
- Schäfer, P., & Leser, U. (2023). WEASEL 2.0: a random dilated dictionary transform for fast, accurate and memory constrained time series classification. *Machine Learning*, 112(12), 4763-4788. <https://doi.org/10.1007/s10994-023-06395-w>
- Secretaría de la Defensa Nacional. (2025). *Estrategia Balam de Seguridad en Carreteras de la Guardia Nacional*. Gobierno de México. Consultado el 5 de marzo de 2026, desde <https://www.gob.mx/defensa/videos/estrategia-balam-de-seguridad-en-carreteras-de-la-guardia-nacional>
- SESNP. (2026). *Incidencia delictiva del Fuero Común 2025* (inf. téc.). Secretariado Ejecutivo del Sistema Nacional de Seguridad Pública. https://www.gob.mx/cms/uploads/attachment/file/1008570/CNSP-Delitos-2025_ene26.pdf
- Shannon, C. E. (1949). Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1), 10-21. <https://doi.org/10.1109/JRPROC.1949.232969>
- Taha, R. A., Youssif, A. A. H., & Fouad, M. M. (2024). Transfer learning model for anomalous event recognition in big video data. *Scientific Reports*, 14. <https://doi.org/10.1038/s41598-024-78414-2>
- Tan, C. W., Dempster, A., Bergmeir, C., & Webb, G. I. (2022). MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36(5), 1623-1646. <https://doi.org/10.1007/s10618-022-00844-1>
- Tang, W., Long, G., Liu, L., Zhou, T., Blumenstein, M., & Jiang, J. (2022). Omni-Scale CNNs: a simple and effective kernel size configuration for time series classification. *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=PDYs7Z2XFGv>
- Tatsunami, Y., & Taki, M. (2022). Sequencer: Deep LSTM for Image Classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 38204-38217.
- Wang, L., Ding, C., Zhang, Y., Zhou, T. H., Ding, W., Ryu, K. H., & Nam, K. W. (2024). Video Abnormal Behavior Recognition and Trajectory Prediction Based on Lightweight Skeleton Feature Extraction. *Sensors*, 24. <https://doi.org/10.3390/s24123711>
- Wang, T., Chen, K., Chen, G., Li, B., Li, Z., Liu, Z., & Jiang, C. (2024). GSC: A Graph and Spatio-Temporal Continuity Based Framework for Accident Anticipation. *IEEE Transactions on Intelligent Vehicles*, 9, 2249-2261. <https://doi.org/10.1109/TIV.2023.3257169>
- Wang, Y., Wu, H., Dong, J., Liu, Y., Wang, C., Long, M., & Wang, J. (2025). Deep Time Series Models: A Comprehensive Survey and Benchmark. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2407.13278>
- Wojke, N., Bewley, A., & Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP)*, 3645-3649. <https://doi.org/10.1109/ICIP.2017.8296962>
- Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2023). TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=ju%5C_Uqw384Oq
- Xu, Z., & Lu, Y. (2023). Abnormal behavior detection algorithm based on multi-branch convolutional fusion neural network. *Multimedia Tools and Applications*, 82, 22723-22740. <https://doi.org/10.1007/s11042-023-14501-2>

- Yao, Y., Wang, X., Xu, M., Pu, Z., Wang, Y., Atkins, E., & Crandall, D. J. (2022). DoTA: Unsupervised Detection of Traffic Anomaly in Driving Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45, 444-459.
- Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020). BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2636-2645. https://openaccess.thecvf.com/content_CVPR_2020/html/Yu_BDD100K_A_Diverse_Driving_Dataset_for_Heterogeneous_Multitask_Learning_CVPR_2020_paper.html
- Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., & Xu, B. (2022). TS2Vec: Towards Universal Representation of Time Series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8), 8980-8987. <https://doi.org/10.1609/aaai.v36i8.20881>
- Zeng, X., Jiang, Y., Wang, Y., Fu, Q., & Ding, W. (2024). Progressive prediction: Video anomaly detection via multi-grained prediction. *IET Image Processing*, 18, 2568-2583. <https://doi.org/10.1049/ipr2.13117>
- Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., & Eickhoff, C. (2021). A Transformer-based Framework for Multivariate Time Series Representation Learning. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2114-2124. <https://doi.org/10.1145/3447548.3467401>
- Zhan, T., He, Y., Deng, Y., Li, Z., Du, W., & Wen, Q. (2025). Time Evidence Fusion Network: Multi-source View in Long-Term Time Series Forecasting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. <https://doi.org/10.1109/TPAMI.2025.3596905>
- Zhang, J., Feng, L., Zhang, H., Wu, Y., & Dong, Y. (2023). Adacket: ADaptive Convolutional KErnel Transform for Multivariate Time Series Classification. *Machine Learning and Knowledge Discovery in Databases: Research Track*, 14173, 189-204. https://doi.org/10.1007/978-3-031-43424-2_12
- Zhang, X., Zeman, M., Tsiligkaridis, T., & Zitnik, M. (2022). Graph-Guided Network for Irregularly Sampled Multivariate Time Series. *International Conference on Learning Representations (ICLR)*. <https://openreview.net/forum?id=Kwm8I7dU-l5>
- Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-object Tracking by Associating Every Detection Box. *Computer Vision – ECCV 2022*, 1-21. https://doi.org/10.1007/978-3-031-20047-2_1