UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

Identificación de Comportamientos de Conducción Peligrosos en Servicios de Transporte Mediante aprendizaje profundo.

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

PRESENTA

ADRIAN LANDA BUENDIA

DIRECTOR DE TESIS

DR. JOSÉ ALBERTO HERNÁNDEZ AGUILAR

REVISORES:

DR. JAVIER ORTIZ HERNÁNDEZ
DRA. LORENA DÍAZ GONZÁLEZ
DR. MARTÍN GERARDO MARTÍNEZ RANGEL
DR. OUTMANE OUBRAM
DR. JOSÉ ALBERTO HERNÁNDEZ AGUILAR

CUERNAVACA, MORELOS. MAYO, 2026

Facultad de Contaduría, Administración e Informática

Tabla de contenidos

RESUMEN

La conducción distraída representa una de las principales causas de accidentes de tránsito en México, particularmente en servicios de transporte, donde las jornadas prolongadas y el uso de dispositivos móviles incrementan el riesgo. En este contexto, la presente investigación tuvo como objetivo desarrollar e implementar modelos basados en redes neuronales convolucionales (CNN) para la identificación y clasificación de diez actividades comunes de conducción, tanto seguras como distractoras, mediante técnicas de visión por computadora.

Para ello, se utilizaron datos provenientes de la base pública *State Farm Distracted Driver Detection*, complementados con una base de datos propia capturada en condiciones reales de conducción en México. Las imágenes fueron sometidas a un proceso de preprocesamiento para mejorar su calidad y facilitar el análisis. Se evaluaron múltiples arquitecturas de CNN, así como enfoques híbridos que combinaron la extracción de características mediante CNN con clasificadores tradicionales como KNN y Random Forest. Adicionalmente, se implementó el modelo YOLOv8 para tareas de detección basada en regiones.

Como aporte metodológico, se incorporó una etapa de preprocesamiento mediante mapas de profundidad (DepthMap), con el propósito de enriquecer la representación espacial de las imágenes. Asimismo, se realizó un estudio de ablación para analizar el impacto de distintas configuraciones arquitectónicas en el desempeño del modelo.

Los resultados mostraron que la arquitectura CNN estándar alcanzó una exactitud del 96.80%, mientras que la integración de DepthMap incrementó el desempeño hasta 98.55%, superando a los modelos híbridos CNN+KNN (86.67%) y CNN+Random Forest (81.71%), y obteniendo resultados competitivos frente a YOLOv8 (96.40%). El estudio de ablación evidenció que arquitecturas más compactas pueden lograr mayores niveles de precisión con menor costo computacional.

En conclusión, los resultados confirman que las redes neuronales profundas constituyen una técnica adecuada para la identificación automática de actividades de conducción, y que la incorporación de información de profundidad mejora la capacidad de generalización del modelo. La propuesta demuestra viabilidad para su implementación en sistemas de monitoreo inteligente del comportamiento del conductor.

ABSTRACT

Distracted driving represents one of the leading causes of traffic accidents in Mexico, particularly in transportation services, where long working hours and the use of mobile devices increase risk. In this context, the present study aimed to develop and implement models based on Convolutional Neural Networks (CNNs) for the identification and classification of ten common driving activities, both safe and distracting, using computer vision techniques.

To this end, data from the public *State Farm Distracted Driver Detection* dataset were used, complemented by a proprietary dataset collected under real driving conditions in Mexico. The images were subjected to a preprocessing stage to enhance their quality and facilitate analysis. Multiple CNN architectures were evaluated, as well as hybrid approaches combining CNN-based feature extraction with traditional classifiers such as KNN and Random Forest. Additionally, the YOLOv8 model was implemented for region-based detection tasks.

As a methodological contribution, a preprocessing stage based on depth maps (DepthMap) was incorporated to enrich the spatial representation of the images. Furthermore, an ablation study was conducted to analyze the impact of different architectural configurations on model performance.

The results showed that the standard CNN architecture achieved an accuracy of 96.80%, while the integration of DepthMap improved performance to 98.55%, outperforming hybrid models such as CNN+KNN (86.67%) and CNN+Random Forest (81.71%), and achieving competitive results compared to YOLOv8 (96.40%). The ablation study demonstrated that more compact architectures can achieve higher accuracy with lower computational cost.

In conclusion, the results confirm that deep neural networks constitute an effective technique for the automatic identification of driving activities, and that the incorporation of depth information improves the model's generalization capability. The proposed approach demonstrates feasibility for future implementation in intelligent driver behavior monitoring systems.

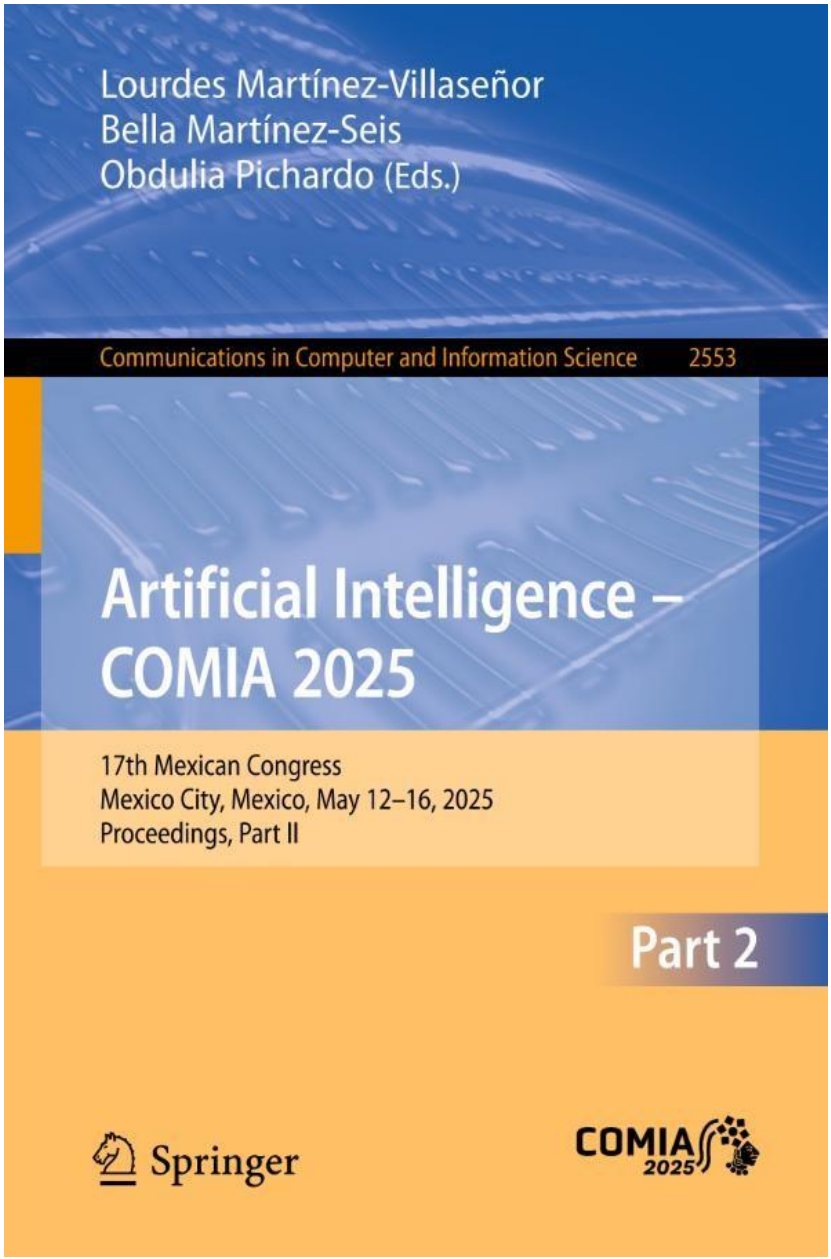
ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ARTICULO PUBLICADO

Landa-Buendía, A., Hernández-Aguilar, J. A., Ortiz-Hernández, J., Díaz-Gónzalez, L., & Oubram, O. (2025). *Identification of dangerous driving patterns through computer vision and deep learning*. En Artificial Intelligence – COMIA 2025: 17th Mexican Congress (Proceedings, Part II) (pp. 3–14). Mexico City, Mexico.

https://doi.org/10.1007/978-3-031-97910-1_1



CAPÍTULO 1. INTRODUCCIÓN

En México, miles de vidas se pierden debido a accidentes que tienen múltiples causas debido a las distracciones al momento de conducir. El incremento en el número de vehículos se correlaciona con un aumento en la incidencia de accidentes INEGI (2020). Cuando se agregan los errores de los conductores, influenciados por dispositivos tecnológicos, la tasa de accidentes se eleva aún más. Los conductores que trabajan para aplicaciones móviles de transporte (por ejemplo: UBER, DIDI), según el Instituto de Seguridad Laboral (2022), se encuentran entre jóvenes y mayores, que son grupos de amplia presencia en este tipo de plataformas, cuyas vulnerabilidades diferenciadas en materia de salud y seguridad en el trabajo han sido sistemáticamente evidenciadas como relevantes, en gran parte debido a extensas jornadas laborales sin pausas adecuadas para descansar o alimentarse. Un factor predominante en la mayoría de los incidentes registrados, son los accidentes que suelen ocurrir debido a distracciones por parte de los conductores. Por ello, se discute un sistema capaz de detectar y clasificar los errores del conductor, proporcionando advertencias oportunas. En este sentido, se realizó una revisión preliminar de modelos de clasificación basados en la extracción de características a través de redes neuronales.

De acuerdo con Olabe (1998). La capacidad del cerebro humano de pensar, recordar y resolver problemas ha inspirado a muchos científicos a intentar o procurar modelar en el ordenador el funcionamiento del cerebro humano. El resultado ha sido una nueva tecnología llamada Computación Neuronal o también Redes Neuronales Artificiales.

Las redes neuronales artificiales proporcionan resultados razonables en aplicaciones donde las entradas presentan ruido o las entradas están incompletas. Algunas de las áreas de aplicación son las siguientes (Olabe, 1998): Análisis y Procesamiento de señales, reconocimiento de imágenes, control de procesos, filtrado de ruido, robótica, procesado del lenguaje, diagnósticos médicos, entre otros.

En esta investigación se pretende analizar los resultados de los modelos basados en redes neuronales convolucionales (CNN) que registran diez actividades de conducción comunes, las cuales se pueden apreciar en la Figura 1.

Figura 1. Comportamientos de conducción comunes.

Así mismo, aplicando redes neuronales convolucionales (CNN) se pretende analizar, clasificar movimientos y comportamientos del conductor, dado que son factores esenciales que pueden afectar la seguridad en la conducción.

1.1 Problemática

La población en México sigue aumentando día a día, al igual que la cantidad de vehículos en circulación. De acuerdo con INEGI (2022), se entiende que la falta de atención del conductor es la causa principal de muchos accidentes automovilísticos.

Tan solo en el año 2022, el número de vehículos registrados alcanzó la cifra de 36,434,606, (INEGI, 2022). La reducción de accidentes y las mejoras en la seguridad pública se han convertido en un objetivo importante de la investigación. En estudios recientes se ha identificado que la postura del conductor, los movimientos, así como la

realización de actividades distractoras durante la conducción, son factores que pueden afectar la conducción segura Al-doori (2021).

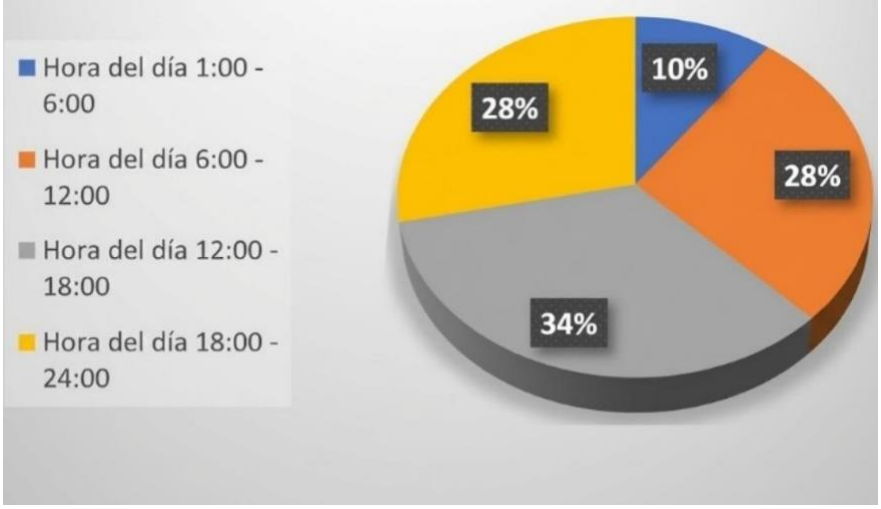


Figura 1.2**.** Horarios en el que ocurren accidentes automovilísticos. (INEGI,2022)

Según el informe del Instituto Nacional de Estadística y Geografía (INEGI) de 2022, los días con la mayor incidencia de accidentes automovilísticos durante la semana son los lunes y sábados (véase la Tabla 1.1), especialmente entre las doce del mediodía y las seis de la tarde (véase figura 1.2). Los factores preponderantes en estos accidentes son las colisiones con vehículos automotores y motocicletas.

Tabla 1.1 Incidencia de accidentes según el día de la semana

Días de la semana	Número de accidentes
Lunes	**68,488**
Martes	50,339
Miércoles	49,758
Jueves	51,486

Viernes	57,177
Sábado	**61,119**
Domingo	54,343
TOTAL	392,710

Por otro lado, la telefonía móvil ha crecido exponencialmente en años recientes y no se estabiliza. Paradójicamente ocupa uno de los primeros lugares como factor detonante en siniestros automovilísticos, pues su uso inapropiado produce distracciones momentáneas que pueden perturbar la conducción del automóvil (Agüero, 2014), especialmente afectan las distracciones de los conductores mientras realizan otras actividades que no tienen que ver con la conducción. Por tal motivo suelen llevar a una menor atención en la carretera y, por lo tanto, aumenta el riesgo de accidentes.

Problema de investigación

Por lo antes descrito, es crucial identificar los comportamientos que generan distracciones en los conductores de transporte con el fin de clasificarlos mediante redes neuronales convolucionales y visión por computadora. Así mismo, se busca probar modelos de clasificación que permitan obtener métricas competitivas, para determinar que actividades de conducción son más frecuentes, y lograr categorizarlas correctamente como acciones peligrosas o comunes.

1.2 Objetivos

General

- Implementar un modelo de clasificación de actividades de conducción basado en el aprendizaje profundo para monitorear y clasificar los comportamientos del conductor.

Específicos

- Construir un conjunto de datos para la identificación de actividades de conducción, integrando información de la base pública State Farm Distracted Driver Detection y una base de datos propia, garantizando su calidad mediante técnicas de preprocesamiento.
- Desarrollar y entrenar modelos basados en redes neuronales convolucionales (CNN) para la identificación y clasificación de diez actividades de conducción, tanto seguras como distractoras.
- Evaluar y comparar el desempeño de distintas arquitecturas de aprendizaje profundo y enfoques híbridos (CNN + KNN, CNN + Random Forest), utilizando métricas como accuracy, precision, recall y matriz de confusión.
- Analizar el impacto del preprocesamiento mediante mapas de profundidad (DepthMap) en el desempeño del modelo, mediante un estudio comparativo.
- Realizar un estudio de ablación para determinar la configuración arquitectónica óptima en términos de precisión y costo computacional.
- Validar el desempeño del modelo propuesto en condiciones reales de conducción, mediante pruebas con imágenes capturadas en entornos reales en México.

1.3 Justificación

En México, la alarmante tasa de accidentes automovilísticos representa un problema multidimensional que involucra diversos factores, desde el incremento en el parque vehicular hasta la influencia de la tecnología en la conducción. La alta incidencia de accidentes entre conductores que trabajan para aplicaciones móviles de transporte destaca la necesidad de abordar esta problemática de manera integral.

La identificación de los días y horas con mayor incidencia de accidentes, según el informe del INEGI, proporciona información crucial. Los lunes y sábados, entre las doce del mediodía y las seis de la tarde, emergen como periodos críticos, impulsando la necesidad de una atención focalizada en estos momentos específicos.

Esta investigación aspira a contribuir a la mejora de la seguridad vial en México mediante la implementación de tecnologías avanzadas y estrategias preventivas basadas en el análisis detallado de las actividades de conducción que pudieran resultar peligrosas mediante el uso de aprendizaje profundo y visión por computadora.

Varios incidentes han involucrado a conductores de plataformas de transporte privado, entre ellos:

- Reportado en el periódico (El Diario Mx, 2022), un conductor de DiDi perdió el control del vehículo, volcándose en la carretera a Ciudad Juárez, específicamente en el kilómetro 17. Como consecuencia de este incidente, el conductor y dos pasajeras resultaron lesionados.
- De acuerdo con el sitio web de noticias (El Heraldo, 2023), en otro incidente, un conductor de 30 años de la plataforma DiDi perdió el control del volante y colisionó contra un poste en la vialidad Los Nogales.

1.4 Hipótesis

Hipótesis nula (H0)

- La incorporación de mapas de profundidad (DepthMap) en el preprocesamiento no produce una mejora estadísticamente significativa en el desempeño de clasificación de actividades de conducción, en comparación con una CNN estándar basada únicamente en imágenes RGB.

Hipótesis alternativa (H1)

- La incorporación de mapas de profundidad (DepthMap) en el preprocesamiento mejora significativamente el desempeño de clasificación de actividades de conducción, en comparación con una CNN estándar basada únicamente en imágenes RGB.

1.5 Alcances y Limitaciones

Alcances

- Se utilizará la base de datos StateFarm Distracted Driver Detection para el entrenamiento de los modelos.
- Se conformará y realizara una nueva base de datos con cinco sujetos de prueba, donde se registrarán las actividades de conducción. Esta nueva base de datos será utilizada para la prueba del modelo
- Se buscará conseguir información de una semana de prueba para cada vehículo analizado.

Limitaciones

- Los comportamientos del conductor pueden cambiar según la hora del día o eventos específicos. Esto podría dificultar la obtención de resultados.
- La iluminación dentro del vehículo durante jornadas nocturnas, pueden tener un impacto negativo al distorsionar y dificultar la interpretación de las imágenes que capturan el comportamiento del conductor.
- Los modelos de detección basados en redes neuronales pueden cometer errores, especialmente en situaciones complejas como cierto tipo de posturas al momento de conducir.
- El desempeño del modelo en el conjunto de datos propio puede reflejar una ligera disminución en la exactitud debido a diferencias en la calidad y el ángulo de las imágenes

1.6 Estructura de Tesis

El presente documento está organizado en varios capítulos.

- **Capítulo 1** se presenta el contexto general de la investigación, el planteamiento del problema, los objetivos y la justificación del estudio, con el propósito de ofrecer un panorama inicial sobre la temática abordada.
- **Capítulo 2** se expone el estado del arte, donde se revisan trabajos previos y antecedentes relacionados con el reconocimiento de actividades de conducción y el uso de técnicas de aprendizaje profundo. **Capítulo 3** se describe de manera detallada la metodología propuesta para el desarrollo de esta investigación, incluyendo el preprocesamiento de datos, la arquitectura de los modelos y las etapas de entrenamiento.
- **Capítulo 4** se presentan los resultados obtenidos a partir de los experimentos realizados, así como su análisis y discusión.
- **Capítulo 5** se muestran las conclusiones alcanzadas y se proponen algunas líneas de trabajo futuro.

CAPÍTULO 2. ESTADO DEL ARTE

De acuerdo con SmartPanel (2023), el Deep Learning o aprendizaje profundo se define como un algoritmo automático estructurado o jerárquico que emula el aprendizaje humano con el fin de obtener ciertos conocimientos. Destaca porque no requiere de reglas programadas previamente, sino que el propio sistema es capaz de «aprender» por sí mismo para efectuar una tarea a través de una fase previa de entrenamiento.

A su vez, también se caracteriza por estar compuesto por redes neuronales artificiales entrelazadas para el procesamiento de información. Se emplea principalmente para la automatización de análisis predictivos. Los algoritmos que componen un sistema de aprendizaje profundo se encuentran en diferentes capas neuronales compuestas por pesos (números). El sistema está dividido principalmente en 3 capas, de acuerdo con SmartPanel (2023).

Capa de entrada (Input Layer): Está compuesto por las neuronas que asimilan los datos de entrada, como por ejemplo una imagen o una tabla de datos.

Capa oculta (Hidden Layer): Es la etapa de la red que realiza el procesamiento de información y procesan los cálculos intermedios. Cada neurona en esta capa comprende cálculos más complejos, con el propósito de identificar bordes, formas, estructuras propias de la imagen.

Salida (Output Layer): Es el último eslabón de la cadena, y es la red que toma la decisión para la identificación de la imagen o realiza alguna conclusión aportando datos de salida.

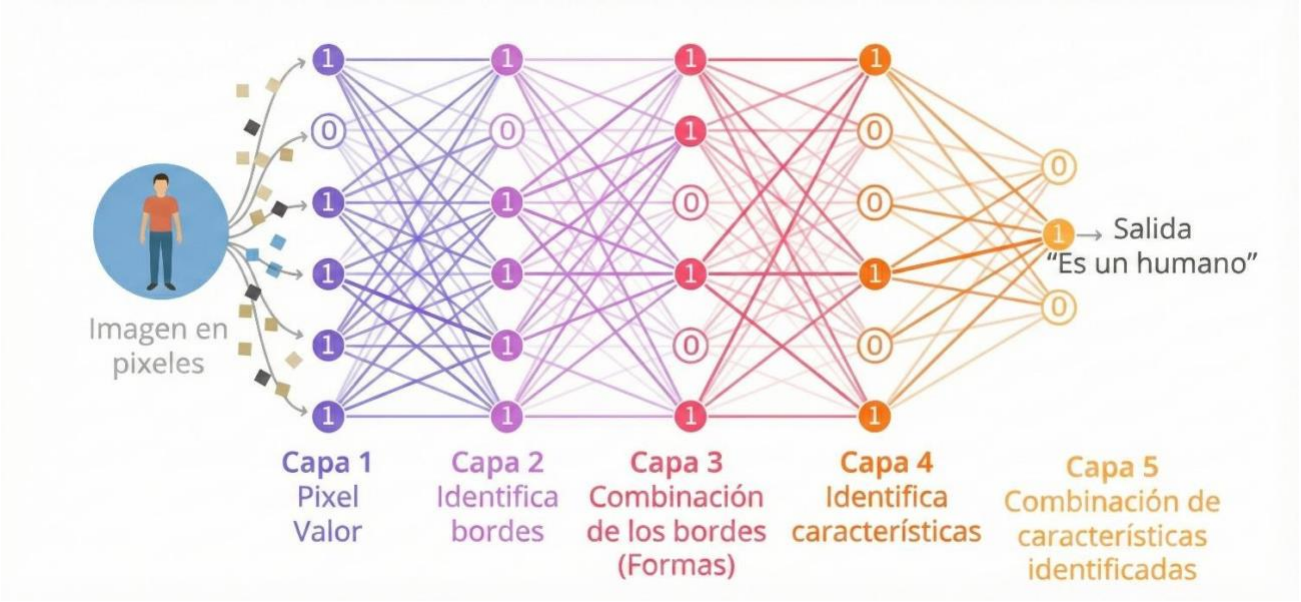


Figura 2.1. Funcionamiento de una red neuronal. (SmartPanel, 2023)

En la figura 2.1, se muestra un modelo de red neuronal para identificar si la imagen que se ingresa como dato de entrada es un perro o no, para lo cual en la primera capa la imagen se trabaja en valores numéricos, en este caso pixeles. Una vez obtenido el valor de cada píxel se envía a diferentes neuronas, para posteriormente pasar a la Capa 2, la cual tiene como objetivo procesar cada uno de los pixeles delimitando los bordes. En la Capa 3, es en donde se combinan los bordes para diseñar formas, y constituir cada uno de los objetos de la imagen. En la Capa 4, se logran identificar las características representativas del objeto (humano), como podría ser una pierna, nariz, ojos, orejas, etc. Como último paso se combinan las características identificadas y se determina mediante la activación de la neurona (0 o 1) si lo anteriormente analizado es semejante al objeto (humano).

2.1 Funcionamiento de las CNNs

Las redes neuronales convolucionales (CNN) son análogas a las redes neuronales artificiales, donde cada neurona recibe un insumo y realiza una operación, la red neuronal seguirá analizando las imágenes utilizando convoluciones que al final, se traducirán en la efectividad de la red para identificar un objeto. En el caso de la clasificación de imágenes el resultado se verá reflejado en la eficiencia de la red para clasificar imágenes nuevas. (Jiménez,2021).

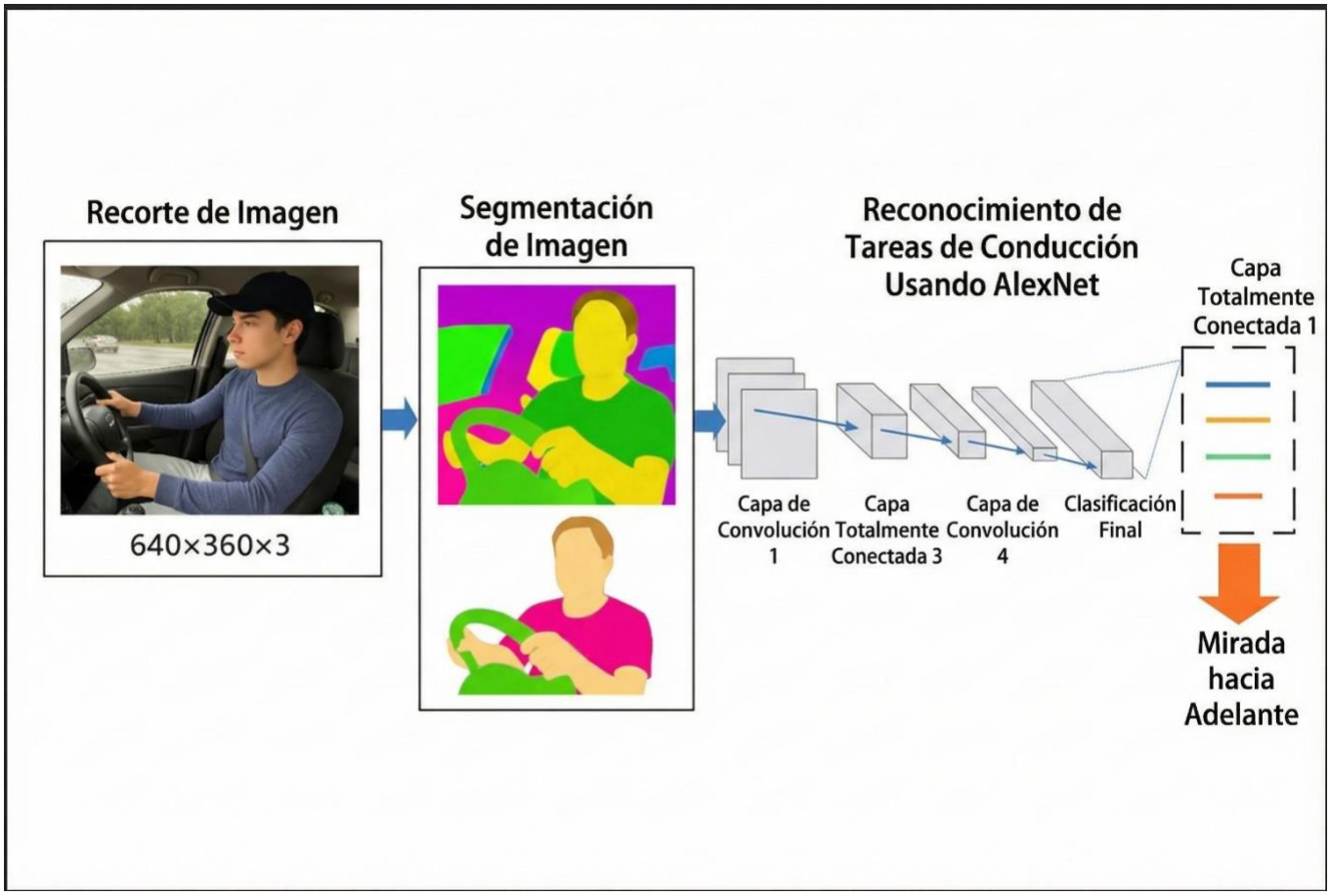


Figura 2.2. Arquitectura del sistema para la identificación de posturas o posiciones peligrosas utilizando la red de convolución AlexNet (Xing et al. 2019).

En la Fig 2.2. Se ilustra el proceso que se lleva a cabo para la clasificación de las imágenes de conductores mediante el uso de una red neuronal convolucional denominada AlexNet, en donde las imágenes originales se almacenan en el formato de 640x360x3. Dichas imágenes en bruto se recortan para acelerar el proceso de entrenamiento de la CNN y aumentar la precisión de la clasificación. Luego, se aplica un algoritmo para segmentar las imágenes y extraer la región del cuerpo del conductor del fondo. Posteriormente se utiliza AlexNet y su procesamiento mediante sus cinco capas de convolución, realizando la extracción de características a diferentes niveles de abstracción, hasta llegar a la última capa, la cual produce como salida la acción identificada.

2.2 Algoritmo k Nearest Neighbors - kNN

Como lo señala Jiménez Alfaro y Díaz Ospina (2022), el aprendizaje automático, es un tipo de inteligencia artificial que aprende o se adapta con el tiempo, este tipo de algoritmo identifica patrones de entrada que van evolucionando de acuerdo con su aprendizaje.

Una de las técnicas de machine learning mayormente utilizadas para la clasificación de imágenes es el algoritmo de k vecinos más cercanos, también conocido como kNN (K nearest neighbors), el cual es un clasificador de aprendizaje supervisado no paramétrico. Para los problemas de clasificación, se asigna una etiqueta de clase sobre la base de un voto mayoritario, es decir, se utiliza la etiqueta que se representa con más frecuencia alrededor de un punto de datos determinado. Como hace mención Sebastian Rashcka (2018), para tareas de clasificación la manera más simple del modelo kNN consiste en predecir la clase objetivo entre lo k ejemplos del entrenamiento más similares a un punto determinado, esta etiqueta de clase puede considerarse como la "moda" de las k etiquetas del entramiento o el resultado de una "votación por mayoría".

Técnicamente se requiere una mayoría superior al 50 %, lo que funciona principalmente cuando solo hay dos categorías. Cuando se tienen varias clases, por ejemplo, cuatro categorías, no se necesita el 50 % de los votos para llegar a una conclusión sobre una clase; dado que se puede asignar una etiqueta de clase con un voto superior al 25 % (IBM, 2023).

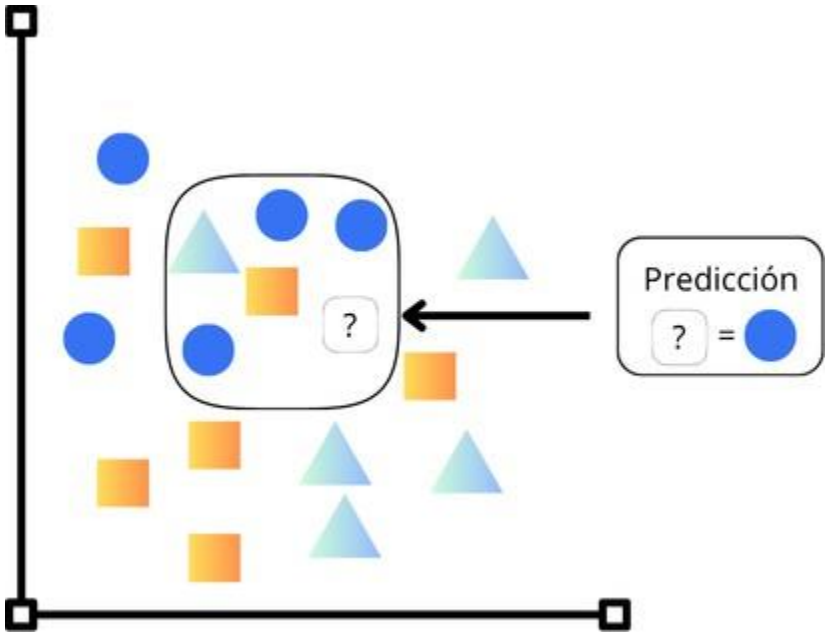


Figura 2.3**.** Ilustración de kNN para un problema de 3 clases con k=5 (Sebastian Rashcka, 2018).

En la Figura 2.3 se ilustra de manera sencilla el funcionamiento del algoritmo kNN, donde el objeto a predecir se clasifica según la cercanía con las instancias más próximas. En el caso de la figura, la clase corresponde a un círculo.

De acuerdo con Raschka (2018), aunque kNN es un aproximador de funciones universal bajo ciertas condiciones, el concepto subyacente es relativamente simple. KNN es un algoritmo de aprendizaje supervisado que simplemente almacena los ejemplos de entrenamiento etiquetados.

2.3 Trabajos relacionados sobre la detección de distracciones del conductor mediante aprendizaje profundo

Xing et al. (2019) proponen un sistema de reconocimiento de actividades del conductor basado en el aprendizaje profundo para monitorear y comprender continuamente los comportamientos del conductor. Los modelos de reconocimiento se entrenan para identificar siete tareas comunes relacionadas con la conducción y también para determinar si el conductor está distraído o no.

Se evalúan tres modelos de CNN diferentes, RestNet50, GoogleNet, AlexNet para el reconocimiento de actividades del conductor y la detección de distracciones. El único sensor utilizado en este estudio es una cámara RGB de bajo costo. Los modelos de CNN toman las imágenes procesadas directamente sin ningún procedimiento manual de extracción de características. Mediante la aplicación del esquema de transferencia de aprendizaje, los modelos de CNN pre entrenados se pueden ajustar de manera eficiente para satisfacer la tarea de detección de comportamientos.

En segundo lugar, se aplica la transferencia de aprendizaje para ajustar finamente los modelos de CNN profundos pre entrenados. Los modelos se entrenan para manejar tanto las tareas de clasificación múltiple como la tarea de clasificación binaria. Por último, se aplica un método de segmentación basado en Gaussian Mixture Model (GMM) no supervisado para procesar las imágenes en bruto y extraer la región del cuerpo del conductor del fondo. Se encontró que al aplicar un modelo de segmentación antes de la red de detección de comportamiento, la precisión de detección del reconocimiento de actividades de conducción puede aumentar significativamente. El resultado de los tres modelos considerando su medida de exactitud (Accuracy) son para AlexNet 91.4%, para GoogleNet 87.5%, y para RestNet del 83.0%.

Al-Doori et al. (2021) utilizan un conjunto de datos de imágenes de acceso público, de 22,424 imágenes. El conjunto de datos utilizado es el de Kaggle Challenge sobre detección de conductores distraídos de State Farm donde se toman diez clases diferentes nombradas de c0 a c9. En este estudio, se utilizó el modelo SqueezeNet previamente entrenado. Realizando un ajuste fino para entrenar este modelo con nuevos datos. El entrenamiento del modelo se realizó adaptando la última capa (flatten10), previo a la capa de la salida Softmax. En este estudio la capa CNN más importante es la capa de convolución. En esta capa, las características de las imágenes se extraen con la ayuda de filtros. Las características obtenidas después de esta capa se denominan mapas de activación. A la capa convolucional le sigue la capa no lineal. Gracias a las capas ocultas con funciones de activación no lineales, los datos que originalmente podrían modelarse de forma lineal se transforman para capturar patrones no lineales; esto permite que las redes neuronales aprendan problemas del mundo real, como los descritos por la capacidad de aproximación universal (IBM,2025).

Los modelos de clasificación probados fueron:

- KNN: Cuando se hace una predicción, se realiza el proceso de clasificación buscando los vecinos más cercanos de los datos recién entrantes.
- SVM (Support Vector Machines): Las operaciones de clasificación se realizan creando tantos hiperplanos como clases existan.
- RF (Random Forest): Se crea un bosque aleatorio, encontrando el nodo raíz y después dividiendo los nodos, proceso que se realiza de forma aleatoria.

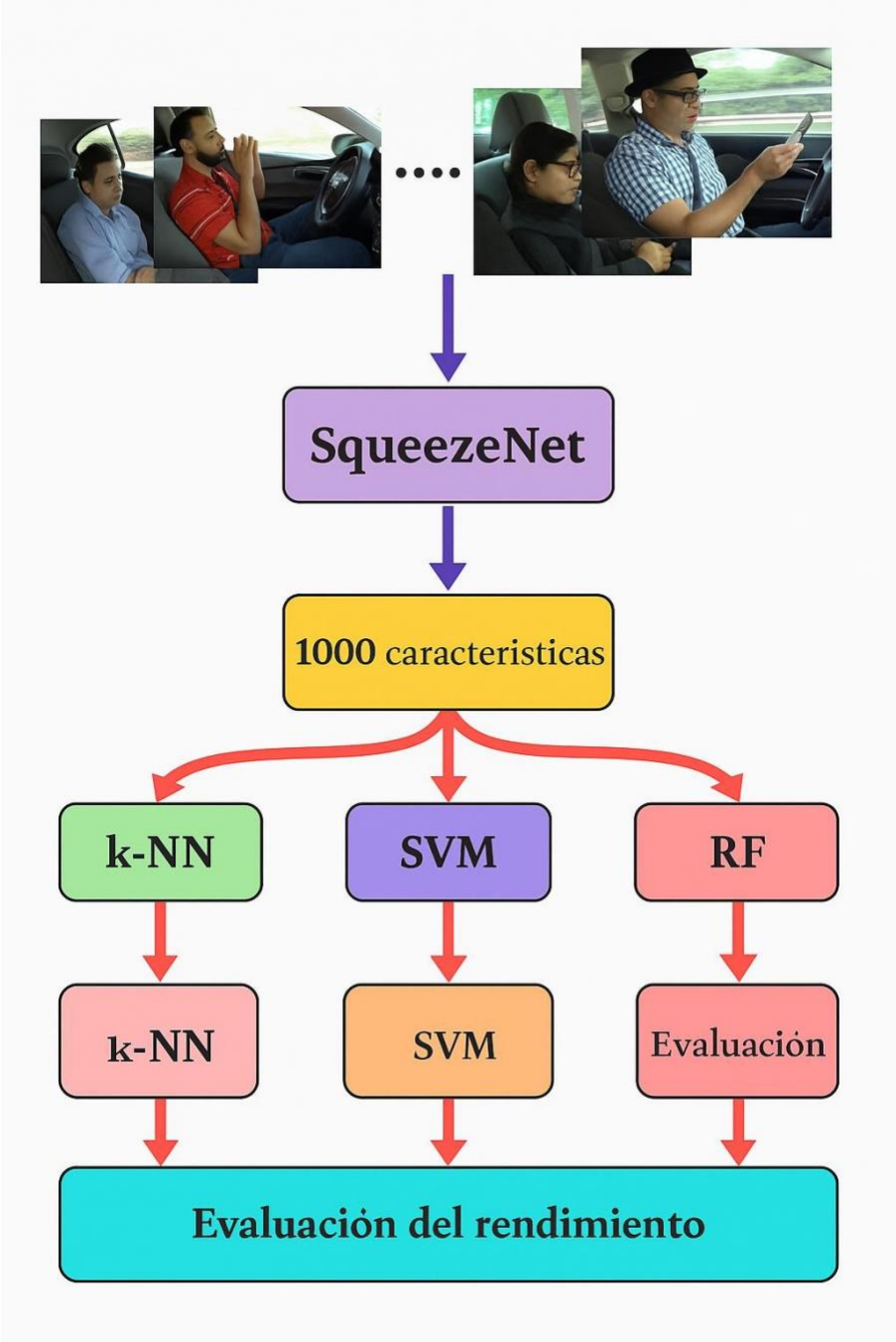


Figura 2.4. Diagrama de flujo del experimento (Al-Doori et al. 2021).

En el diagrama de flujo del experimento de la figura 2.4, comienza con las imágenes de las posturas de conductores, de esta manera la red identifica las características y se extraen con el modelo SqueezeNet. Las propiedades de las imágenes se toman en la capa justo antes de la capa softmax, que es la última del modelo SqueezeNet. Las 1000 características se usan como entrada para los 3 algoritmos. Los autores determinaron que el clasificador k-NN tiene el mayor éxito de clasificación con un 98.1% de accuracy, y el más bajo RF con 88.7%.

Huang et al. (2020) proponen un modelo cooperativo de CNNs, para las características de comportamiento distraído, donde se extraen en paralelo estas características para la clasificación de imágenes, utilizando los tres modelos pre entrenados integrados ResNet50, InceptionV3 y Xception. Estos tres modelos se entrenan en el conjunto de datos procedentes del repositorio ImageNet (Imagenet, 2025), y luego se aplica el aprendizaje por transferencia para realizar un ajuste fino. En segundo lugar, el módulo de concatenación de características se utiliza para fusionar profundamente las características extraídas del módulo cooperativo CNN, generando la entrada para el módulo de clasificación de características.

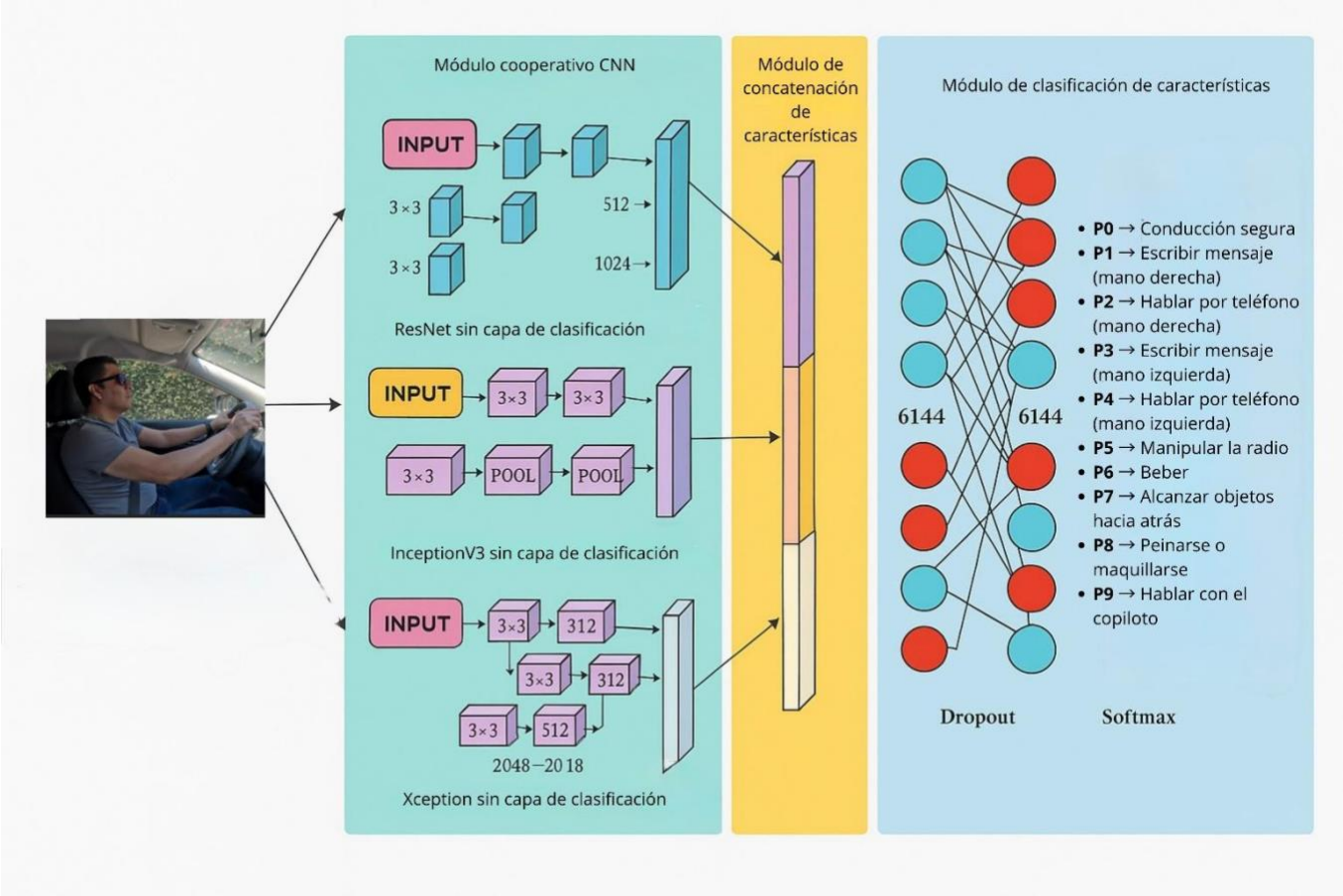


Figura 2.5**.** Arquitectura del marco híbrido CNN (Huang et al, 2020).

Como se ilustra en la Figura 2.5, la arquitectura comienza con el procesamiento de imágenes en su resolución original de 640x480 píxeles. Posteriormente, las imágenes se ingresan al módulo cooperativo CNN, donde cada capa realiza un procesamiento específico. En estas capas se aplican operaciones de convolución, pooling y redimensión, hasta llegar a las capas finales, donde se realiza la concatenación de características. Finalmente, el

modelo completa el proceso con la clasificación de dichas características. Obteniendo un desempeño en la detección de conductas de conducción del 96.74% de accuracy.

En Jimiama et al. (2020) se presenta una novedosa arquitectura de aprendizaje profundo llamada Convolutional-Stacked Long Short-Term Memory (C-SLSTM) para la detección de posturas de distracción del conductor. Una arquitectura de aprendizaje profundo que supera a los modelos CNN de última generación actuales al clasificar posturas de conducción distraídas mediante imágenes estáticas. El modelo consta de redes CNN y BiLSTM concatenadas. Las redes CNN aprenden automáticamente las características espaciales de las imágenes y los LSTM extraen las correlaciones espectrales entre los mapas de características producidos por las CNN. La metodología está motivada por el rendimiento de última generación de los LSTM en la clasificación de imágenes hiperespectrales, donde los LSTM se utilizan para capturar correlaciones entre los canales espectrales de imágenes estáticas. Los mapas de características apiladas pasan por la capa de agrupación, que reduce la resolución de las representaciones de entrada mediante un proceso de discretización basado en muestras. Posteriormente, la capa de activación convierte los datos apilados y reducidos en características específicas dependiendo de la función de activación que se utiliza (por ejemplo, unidad lineal rectificada) o (ReLU), la cual convierte todos los valores negativos a cero y mantiene todos los valores. Véase figura 2.6.

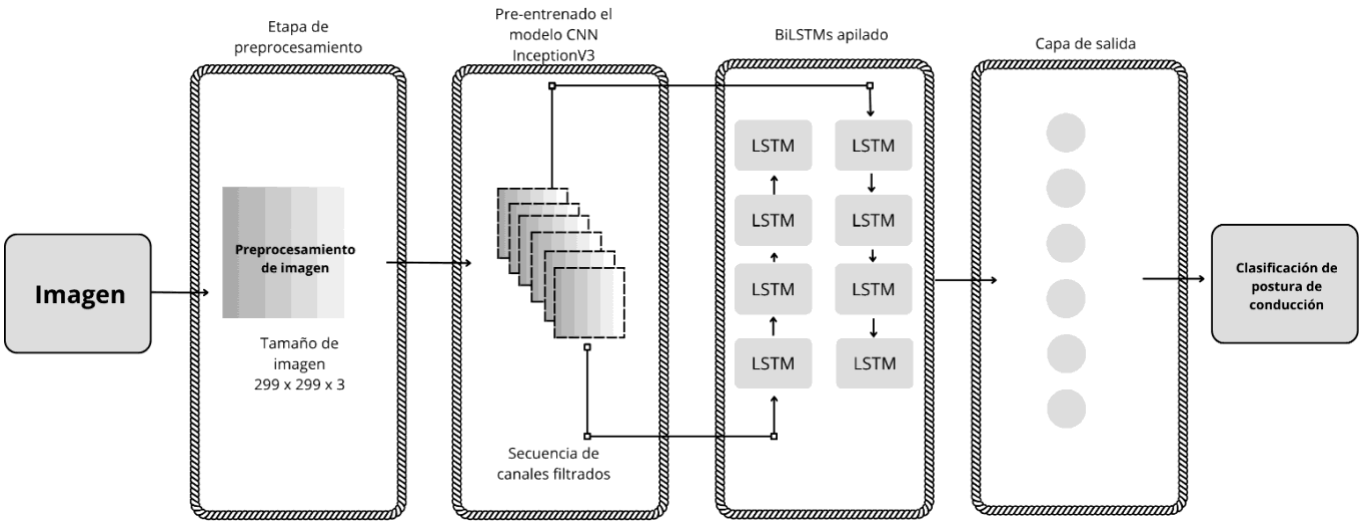


Figura 2.6. Propuesta de la arquitectura CNN-BiLSTM para la detección de posturas (Jimiama et al.,2020).

En Tabla 2.1. se observa la comparación del resultado de su método utilizado para clasificar imágenes con respecto a lo reportado en la literatura. En donde se obtuvo un mayor desempeño al combinar un modelo de red neuronal profunda con BiLSTM, un tipo de red neuronal recurrente dando una accuracy del 92.70%.

Tabla 2.1. Comparación de resultados en la clasificación de distracciones de conducción (Jimiama et al., 2020).

Modelo	**Pérdida (Loss)**	**Accuracy (%)**
VGG-16	1.2466	76.13
ResNet50	0.6615	81.69
Ensamble de InceptionV3 con GA-Weighted algorithm	0.64	90.06
InceptionV3	0.5723	84.41
InceptionV3-LSTM	0.4445	89.82
C-SLSTM	**0.2793**	**92.70**

Aljasim y Kashef (2022) diseñaron un sistema de detección de distracciones basado en conjuntos, concretamente E2DR, para mejorar la precisión de la detección de distracciones del conductor y proporcionar recomendaciones. El modelo Ensemble/based Driver Distraction with Recommendations (E2DR) es la combinación de dos o más modelos de detección para proporcionar predicciones mejores y más precisas que los modelos individuales. Este modelo mejora la generalización del proceso de detección para evitar el sobreajuste. Como se muestra en la Figura 2.7. el conjunto en la arquitectura E2DR resulta de la combinación de dos modelos básicos.

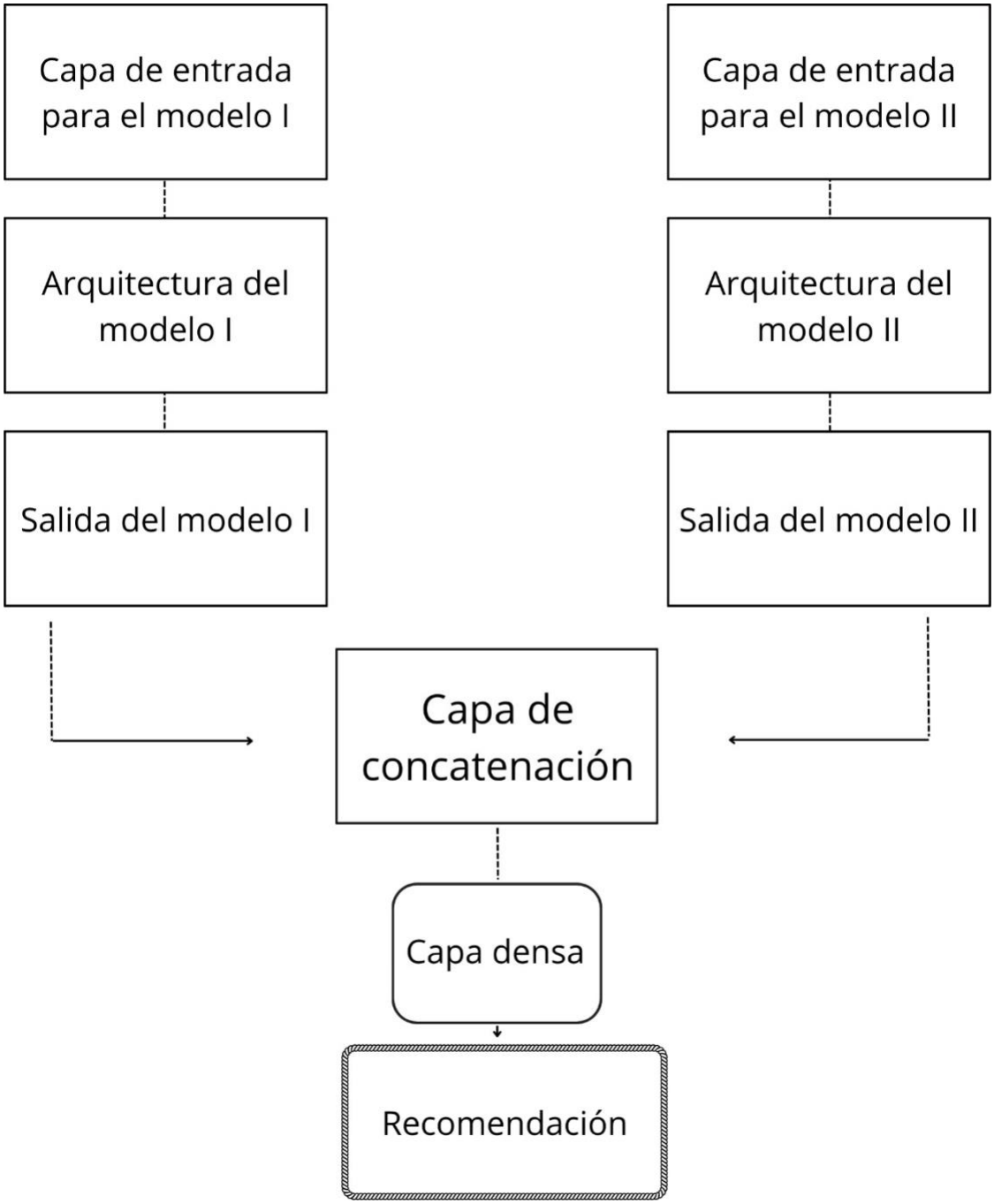


Figura 2.7. Arquitectura EDR2 (Alijasam y Kashef, 2022).

Los modelos adoptados en este artículo se encuentran entre los modelos CNN de mayor rendimiento. Para todos los modelos, se utiliza una tasa de aprendizaje de 0,001 y una entropía cruzada categórica como función de pérdida. Al igual se desarrollan seis modelos utilizando variantes del modelo base 1 y del modelo base 2, como E2DR (A1, A2) donde A1 y A2 ∈ {ResNet, VGG16, MobileNet, Inception}. El modelo E2DR combina 2 de los modelos

mencionados utilizando el método de conjunto de generalización apilada (SG-Stacked Generalization). El método de conjunto SG utiliza los resultados de los modelos base previamente entrenados, los concatena y los envía a una capa densa para clasificación. Una vez clasificado el comportamiento distractor, se proporciona un conjunto de acciones recomendadas para garantizar la seguridad. Finalmente, se calcula una medida de evaluación para cada modelo de E2DR (A1, A2) utilizando las métricas Accuracy, Loss, F1, Precision y Recall.

En la Tabla 2.2. se muestra el rendimiento de los modelos de clasificación funcionando de manera individual. En ella se observa el modelo predominante con mayor porcentaje de exactitud en el entrenamiento, el cual resultó ser VGG16, así mismo, para la validación y para el test, el que obtuvo las mejores métricas fue el modelo de ResNet.

Tabla 2.2. Rendimiento individual de los modelos de clasificación de imágenes (Alijasam y Kashef, 2022).

Modelo	**Accuracy Entrenamiento**	**Accuracy Validación**	**Accuracy en Test**
ResNet	0.89	0.88	0.88
VGG16	0.94	0.86	0.87
Mobile Net	0.88	0.84	0.82
Inception	0.83	0.84	0.83

Al observar la tabla 2.3, se puede identificar el rendimiento de los modelos en conjunto. RestNet50 - VGG16 encabeza los mayores puntajes obtenidos de las métricas, demostrando ser el mejor método para la clasificación en este estudio.

Tabla 2.3. Rendimiento en conjunto de los modelos de clasificación de imágenes (Alijasam y Kashef, 2022).

Modelo E2DR	Accuracy	Precision	Recall	F1 Score	Loss
MobileNet-Inception	0.88	0.89	0.88	0.88	0.55
RestNet50-Inception	0.88	0.89	0.88	0.88	0.47
ResNet50-MobileNet	0.90	0.91	0.9	0.9	0.43
VGG16-Inception	0.90	0.91	0.9	0.9	0.39
VGG16-MobileNet	0.91	0.92	0.91	0.91	0.42
ResNet50-VGG16	**0.92**	**0.92**	**0.92**	**0.92**	**0.37**

En Koay et al. (2021) se presenta un conjunto de ResNets, denominado Optimally-weighted Image-Pose Approach (OWIPA) (Enfoque de pose de imagen con ponderación óptima), para clasificar la distracción a través de imágenes originales y de estimación de pose. Las imágenes de estimación de pose se generan a partir de HRNet y ResNet, usando ResNet101 y ResNet50.

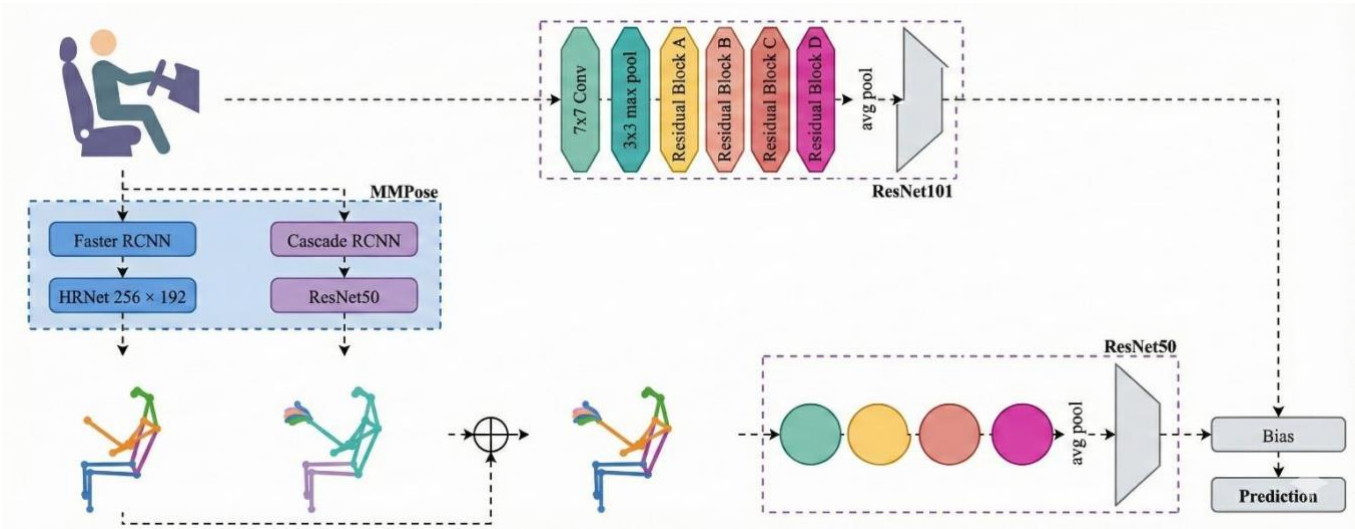


Figura 2.8. Arquitectura del modelo de clasificación de la estimación de pose y detección de distracciones (En Koay et al. 2021).

En primera instancia las imágenes originales se preprocesan, en la segunda etapa, las imágenes escaladas se someten a estimaciones de la postura del cuerpo y de las manos a través de dos redes diferentes, HRNet y ResNet50. En donde se entrena a ResNet101 y ResNet50 para clasificar las imágenes originales y las imágenes de estimación de pose, respectivamente. El primer modelo ResNet se utiliza para clasificar en función de las imágenes originales, que se escalan a 360×360. La estimación de la postura del cuerpo se realiza a través de HRNet. Antes de introducir HRNet, se utiliza Faster RCNN para detectar a la persona y reducir el espacio de búsqueda y el tiempo de cálculo. Y Cascade RCNN se utiliza para detectar la mano. Una vez que se completan las estimaciones de la pose del cuerpo y de la mano, los resultados se suman para formar las imágenes de la pose completa.

Tabla 2.4. ResNet50 y ResNet101 utilizados en el método propuesto, con imágenes de entrada redimensionadas (Koay et al. 2021).

Capa	**Tamaño de salida**	**ResNet50**	**ResNet101**
Entrada	360 x 360 x 3	Imagen de entrada	
Convolución	180 x 180 x 64	7 x 7, 64 stride 2	
Max Pool	90 x 90 x 64	3 x 3 max pool, stride 2	
Bloque Residual	90 x 90 x 256	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$
Bloque Residual	45 x 45 x 512	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$
Bloque Residual	23 x 23 x 1024	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 23$
Bloque Residual	12 x 12 x 2048	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$
Avg pool	1 x 1 x 4096	Average pool, Fully connected	
Flatten	4096	Flatten, BatchNorm	
Dropout	512	Dropout, $p = 0.25$, ReLU, BatchNorm	
Output	10	Dropout, $p = 0.5$, Linear	

Como se evidencia en la tabla 2.4. Los siguientes parámetros fueron utilizados para la predicción de los modelos Convolution, MaxPooling, Residual Block, Dropout. Ambas arquitecturas inician con una capa de convolución seguida de una operación de max pooling, que reduce la dimensionalidad de la imagen. Posteriormente, se estructuran en bloques residuales que permiten un entrenamiento más profundo. Finalmente, las arquitecturas concluyen con una etapa de average pooling, seguida una capa flatten, normalización (BatchNorm), regularización mediante dropout, y una capa de salida que realiza la clasificación en 10 clases.

Loyola Ayque (2024) desarrolló un sistema de reconocimiento de comportamiento de conducción distraída utilizando una red profunda VGG16 preentrenada combinada con técnicas de aprendizaje por currículo (*Curriculum Learning*), en donde el modelo primero aprende con ejemplos fáciles de baja complejidad y gradualmente se enfrenta a ejemplos más difíciles. El objetivo principal fue mejorar la precisión en la detección de conductas distractoras como el uso del teléfono, comer o conversar con pasajeros, mediante la identificación de las imágenes. Para ello, se usó la base de datos *State Farm Distracted Driver Detection*, compuesta por más de 22,000 imágenes etiquetadas en diez clases (c0–c9), que representan diferentes comportamientos de conducción. Las imágenes fueron preprocesadas con normalización y redimensionadas a 256×256 píxeles. En la metodología realizaron transferencia de aprendizaje (*transfer learning*) y ajuste fino (*fine-tuning*) sobre la arquitectura VGG16, aplicando un esquema de aprendizaje por currículo en tres fases progresivas (fácil, intermedia y difícil) con distintas configuraciones de entrenamiento (2-3-4, 4-3-2 y 3-3-3 épocas por fase). El modelo fue evaluado mediante métricas de precisión (*accuracy*), mostrando resultados superiores al 98.47% de exactitud, superando ampliamente el rendimiento de la versión tradicional.

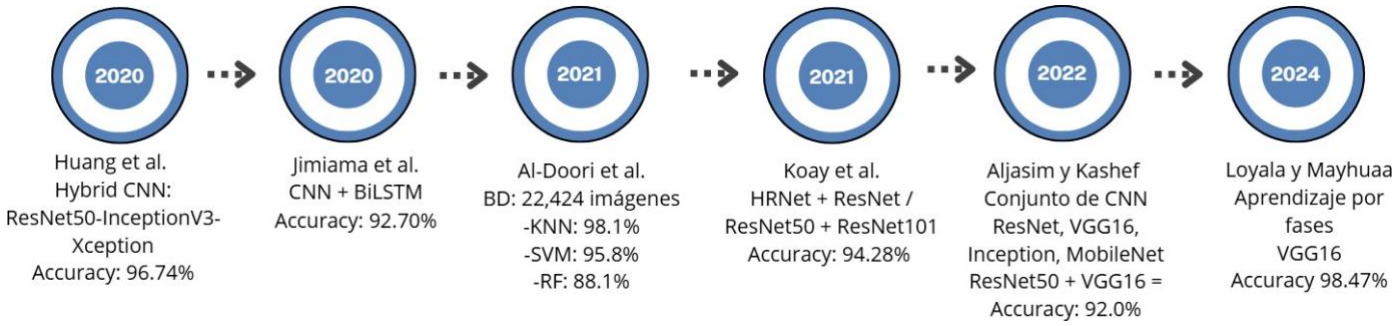


Figura 2.9. Línea de tiempo de los trabajos más influyentes relacionados con este proyecto.

La figura 2.9 muestra una línea del tiempo de acuerdo con el estado del arte revisado, en ella se ven los trabajos más relevantes utilizados para este proyecto. Se observa que los trabajos utilizan diversas arquitecturas para la detección de conductores distraídos, como redes híbridas (Huang et al., 2020), arquitecturas recurrentes (Jimiama

et al., 2020) y combinaciones de CNN con clasificadores tradicionales (Al-Doori et al., 2021). Esto tiene una relación con nuestro proyecto ya que en la presente investigación se combinan arquitecturas de redes convolucionales con algoritmos de clasificación (k-NN y RF), los cuales se aplican a un contexto local y se evalúan con la base de datos pública StateFarm Distracted Driver Detection. Haciendo el uso de un preprocesamiento y técnicas como el DepthMap. Complementado con un estudio de ablación que permite evaluar la exactitud de cada arquitectura, al igual que su eficiencia computacional.

Brecha científica

A pesar de los avances en la identificación automática de comportamientos de conducción mediante modelos de visión por computadora y aprendizaje profundo, la mayoría de los enfoques actuales se basan exclusivamente en información visual RGB, limitando la capacidad de los modelos para capturar características espaciales más profundas asociadas a la postura y contexto del conductor.

Asimismo, si bien existen arquitecturas avanzadas como redes neuronales convolucionales profundas y modelos de detección como YOLO, pocos trabajos exploran la incorporación de información adicional, como mapas de profundidad, en tareas de clasificación de actividades de conducción. De igual forma, es limitada la evidencia sobre el impacto de este tipo de preprocesamiento en la mejora del desempeño de los modelos, particularmente en escenarios reales de conducción.

En este sentido, la presente investigación aborda esta brecha mediante la incorporación de mapas de profundidad (DepthMap) como etapa de preprocesamiento, con el objetivo de enriquecer la representación espacial de las imágenes y mejorar el desempeño de los modelos de clasificación. Adicionalmente, se realiza un estudio comparativo entre distintas arquitecturas y enfoques híbridos, así como un estudio de ablación, lo que permite analizar de manera sistemática la contribución de cada componente del modelo.

CAPÍTULO 3. METODOLOGÍA DE LA INVESTIGACIÓN

En esta sección se presenta la metodología desarrollada para este proyecto de investigación. En el capítulo anterior se revisó con el estado de arte, las cuales se utilizó para comprender el funcionamiento de las redes neuronales artificiales implementadas en la identificación de imágenes en posturas de conducción. Y proponer la siguiente metodología Fig 3.1.

Figura 3.1. Metodología de trabajo.

En la presente investigación se llevó a cabo un análisis de los modelos de redes neuronales convolucionales más utilizados, así como de los algoritmos de clasificación con mejor desempeño reportados en la literatura, con el propósito de seleccionar aquellos que ofrecieran resultados óptimos. A partir de este análisis se determinó la implementación de You Only Look Once (YOLO) en su versión 8 como red neuronal profunda, complementado con el desarrollo de un algoritmo propio basado en el estado del arte.

Posteriormente, se aplicaron técnicas de preprocesamiento orientadas a mejorar la calidad y representación de las imágenes, entre las que destacó el uso de DepthMap para optimizar la detección de características relevantes. Finalmente, se incorporaron diferentes algoritmos de clasificación en las capas finales de la red, tales como el método supervisado k-nearest neighbors (k-NN), Random Forest y Support Vector Machine (SVM), con el objetivo de comparar su desempeño y seleccionar la configuración más adecuada

3.1 Base de datos

Con el conocimiento sobre la clasificación, se consultaron bases de datos de acuerdo con la problemática, la cual consiste en actividades de conducción peligrosas en servicios de transporte. Se seleccionó un repositorio de

imágenes perteneciente a State Farm, “Distracted Driver Detection” (State Farm, 2016), que cuenta con veintidós mil cuatrocientos veinticuatro imágenes. Al identificar una base de datos adecuada para este proyecto, también se buscó recopilar imágenes propias para realizar la prueba de los modelos que serán utilizados. Esta segunda base de datos será recopilada de personal que se dedica al servicio de transporte. Las imágenes se obtendrán con un dispositivo móvil que se encontrará colocado estratégicamente dentro del vehículo con un soporte ergonómico, el cual permite realizar la toma de fotografías durante un trayecto de conducción.

Se utilizó la base de datos StateFarm Distracted Driver Detection para el entrenamiento de los modelos. (Kaggle, s. f.), las imágenes se encuentran clasificadas en actividades de conducción seguras y peligrosas divididas en diez diferentes clases, C0: Conducción segura, C1: Mensajes de texto-mano derecha, C2: Hablando por teléfono-mano derecha, C3: Mensajes de texto-mano izquierda, C4: Hablando por teléfono-mano izquierda, C5: Operar la radio, C6: Beber-Comer, C7: Mirar hacia atrás, C8: Peinando-Maquillando, C9: Hablando con el pasajero.

La estructura de los datos se identificó de la siguiente manera:

- **Etiquetado:** Las imágenes están etiquetadas con la clase correspondiente a la actividad del conductor.
- **Formato de imagen:** Las imágenes se encuentran en formato JPG de alta resolución.
- **Número de imágenes por clase:**
    - C0: 2,487 imágenes
    - C1: 2,269 imágenes
    - C2: 2,265 imágenes
    - C3: 2,224 imágenes
    - C4: 2,267 imágenes
    - C5: 2,268 imágenes
    - C6: 2,279 imágenes
    - C7: 2,282 imágenes
    - C8: 2,273 imágenes
    - C9: 2,325 imágenes

    Como se puede observar el número de imágenes por clase es muy parecido por lo que no se utilizó balanceo de clases.

3.2. Adquisición de imágenes para la prueba del modelo.

La adquisición de imágenes para la etapa de prueba se realizó mediante un dispositivo móvil que captura posturas de conducción a través de grabaciones en video. En la Figura 11 se observa dicho dispositivo, colocado en un soporte ergonómico estratégicamente situado en la parte superior derecha del parabrisas del vehículo. Se generaron tres videos de 10 minutos, donde se registraron las actividades de manejo del conductor A partir de los videos grabados se extrajeron 2,528 fotogramas para conformar el conjunto de datos de prueba.

Figura 3.2. Dispositivo de captura de imágenes

Capturando poses similares a la base de datos Distracted driver detection, siendo clasificadas estas imágenes en las categorías de la C0 a C9.

3.3 División de los datos

Al contar con dos repositorios se comenzará con la prueba de los modelos de redes en la clasificación de actividades de conducción. Para ello se hará uso de la primera base de datos de StateFarm para el entrenamiento del modelo ajustando sus parámetros, para posteriormente realizar la validación, en donde con un pequeño subconjunto de los datos que no se usa directamente en el entrenamiento del modelo, será utilizado en la validación para ajustar los hiperparámetros y evitar un sobreajuste. De esta manera, finalizaríamos con la prueba del modelo haciendo uso de nuestro propio repositorio de datos, con el propósito de probar con un conjunto de

datos que es desconocido para la red neuronal. Siendo utilizado para evaluar el modelo después del entrenamiento y validación.

Entrenamiento del modelo (80%):

Para la fase de entrenamiento se utilizó la base de datos StateFarm Distracted Driver Detection, la cual fue dividida siguiendo lo reportado en la literatura, de tal manera que permitiera garantizar la representación balanceada de las clases. En esta etapa, el conjunto de datos de entrenamiento se empleó con el propósito de enseñar al modelo a reconocer patrones visuales relacionados con las diferentes posturas y actividades de conducción. El total de imágenes destinadas al entrenamiento fue de 17,450 las cuales permitieron que la red neuronal ajustara sus pesos internos mediante un proceso iterativo de retropropagación y optimización.

Validación de datos (10%):

El conjunto de validación corresponde a un subconjunto de datos que no participa directamente en el entrenamiento del modelo. Su función principal es proporcionar una métrica de desempeño durante el ajuste de los hiperparámetros, tales como la tasa de aprendizaje, número de capas, neuronas por capa y funciones de activación (Ilemobayo et al., 2024) Este procedimiento ayuda a evitar el sobreajuste (overfitting) y permite seleccionar la arquitectura más adecuada. En total, el conjunto de validación estuvo compuesto por 2,485 imágenes.

Conjunto de prueba (10%)

Finalmente, se conformó un conjunto de prueba independiente y completamente separado de los datos de entrenamiento y validación. Este conjunto tiene como objetivo evaluar el rendimiento final del modelo bajo condiciones similares a un escenario real, es decir, midiendo su capacidad de generalización frente a datos que no han sido utilizados previamente en el proceso de aprendizaje. El conjunto de prueba estuvo constituido por 2,528 imágenes, provenientes de la base de datos complementaria adquirida mediante capturas en video y extracción de fotogramas.

3.4 Preprocesamiento

Una vez obtenida una gran cantidad de imágenes se llevará a cabo un procedimiento de preprocesamiento, el cual consiste en realizar una limpieza de las muestras obtenidas, un etiquetado de las posturas de conducción de

acuerdo con las diez clases, un ajuste en el formato y dimensiones de las imágenes, y por último un balanceo de clases.

3.4.1 Ajuste de dimensiones

Las imágenes provenientes de las bases de datos fueron sometidas a un proceso de redimensionamiento con el fin de optimizar los recursos disponibles y reducir el costo computacional durante el entrenamiento y la validación de los modelos. Este procedimiento permitió estandarizar las dimensiones de todas las imágenes, evitando variaciones en el tamaño que pudieran afectar el desempeño de las redes neuronales convolucionales. Cada imagen fue convertida a un formato de 150 x 150 píxeles, lo que garantizó un equilibrio en las características visuales relevantes y la eficiencia en el procesamiento de la gran cantidad de imágenes.

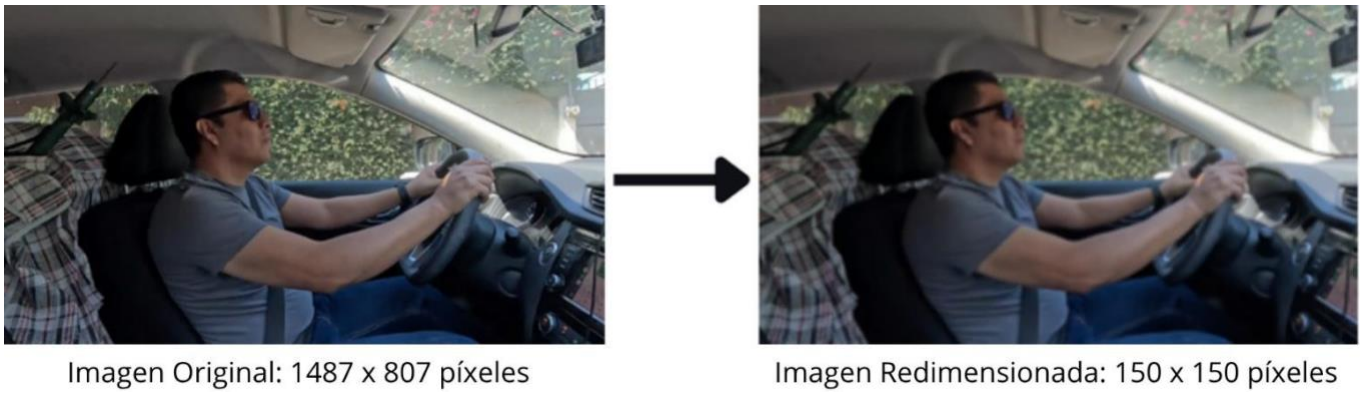


Figura 3.3. Redimensión de imágenes.

Para mejorar la capacidad de generalización del modelo y prevenir el sobreajuste, se implementaron técnicas de preprocesamiento y aumento de datos mediante la clase ImageDataGenerator de Keras. Este procedimiento permitió incrementar artificialmente la diversidad del conjunto de entrenamiento. Como se aprecia en la figura 3.4.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,              # Normalizar imágenes
    shear_range=0.2,             # Transformación aleatoria
    zoom_range=0.2,              # Zoom aleatorio
    horizontal_flip=True)        # Volteo horizontal de las
```

Figura 3.4. Código de preprocesamiento de imágenes.

- Normalización de imágenes: Cada píxel de la imagen se re escaló a un rango entre 0 y 1, dividiendo los valores originales (0–255) entre 255. Esta normalización facilita la convergencia del modelo durante el proceso de optimización.
- Transformación aleatoria (shear): Se aplicó un valor de shear_range=0.2, lo que generó ligeras deformaciones angulares en las imágenes, simulando variaciones de perspectiva.
- Zoom aleatorio: Con un zoom_range=0.2, las imágenes fueron ampliadas o reducidas de manera aleatoria, lo que permitió al modelo ser más robusto frente a cambios de escala.
- Volteo horizontal: Se activó la opción horizontal_flip=True, la cual invierte las imágenes horizontalmente. Esto resulta útil en el reconocimiento de posturas, ya que permite representar escenarios simétricos (por ejemplo, movimientos de manos en lados opuestos).

Estas técnicas de aumento de datos contribuyeron a generar un conjunto de entrenamiento más variado, sin necesidad de recopilar nuevas imágenes, mejorando así la capacidad de generalización del modelo en escenarios reales.

3.5 Entrenamiento y prueba de los modelos

Una vez finalizado el preprocesamiento de las imágenes, se procedió a la implementación de diversas arquitecturas de redes neuronales para la clasificación y detección de patrones. En primer lugar, se evaluó la arquitectura propuesta desarrollada específicamente para este proyecto, esta se diseñó con múltiples capas convolucionales y capas densas que permiten extraer características relevantes de las imágenes. Véase figura 3.5. Posteriormente, se exploró una variante de esta misma arquitectura, en la cual se reemplazó la última capa por un algoritmo de clasificación adicional (KNN o Random Forest). Comenzando con la primera arquitectura la cual se muestra en la Figura 3.5.

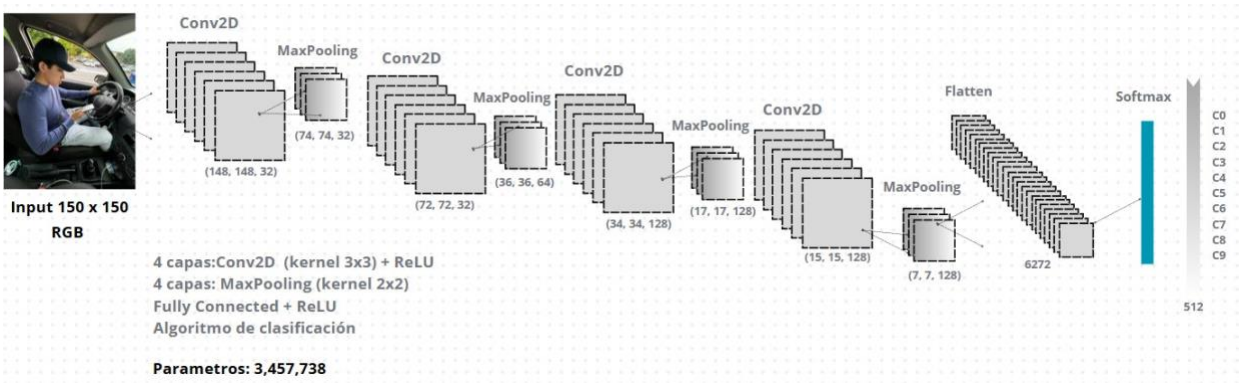


Figura 3.5**.** Arquitectura de red neuronal profunda.

La arquitectura propuesta está compuesta por un total de 9 capas, cuidadosamente diseñadas para extraer y procesar características relevantes de las imágenes. En primer bloque, cuenta con 4 capas convolucionales, encargadas de identificar patrones locales mediante filtros que capturan características como bordes, texturas y formas. Cada una de estas capas convolucionales es seguida por una capa de MaxPooling, cuyo objetivo es reducir la dimensionalidad de los mapas de características, disminuir el costo computacional y proporcionar invariancia frente a pequeñas traslaciones en la imagen.

Tras las capas convolucionales y de pooling, se incorporó una capa de aplanamiento (flatten), que transforma los mapas de características multidimensionales en un vector unidimensional, permitiendo su conexión con las capas densas o de salida del modelo. Finalmente, se aplican las funciones de activación correspondientes en las capas finales, las cuales determinan la respuesta de cada neurona y permiten que el modelo aprenda relaciones no lineales entre las características extraídas y las clases de salida. Esta combinación de capas convolucionales, de pooling y de aplanamiento proporciona un equilibrio entre la extracción de características complejas y la eficiencia computacional, asegurando un desempeño robusto en tareas de clasificación de imágenes.

En total, la arquitectura contó con 3,457,738 parámetros entrenables, y las pruebas fueron realizadas en un entorno de cómputo compuesto por un procesador Intel Core i7-14700F, 32 GB de memoria RAM y una GPU NVIDIA RTX 4070 Super con 12 GB de VRAM, sobre Windows 11, lo que permitió llevar a cabo el entrenamiento y la validación de los modelos de manera eficiente y en tiempos razonables.

3.6 Entrenamiento del modelo

Se comenzó con las pruebas del entrenamiento, en esta etapa, se llevó a cabo el entrenamiento de la red neuronal convolucional utilizando el conjunto de datos previamente procesado.

```
Epoch 1/50
545/545 [==============================] - 171s 314ms/step - loss: 1.1629 - accuracy: 0.6023 - val_loss: 0.3766 - val_accuracy: 0.8782
Epoch 2/50
545/545 [==============================] - 133s 244ms/step - loss: 0.3262 - accuracy: 0.9004 - val_loss: 0.1278 - val_accuracy: 0.9627
Epoch 49/50
545/545 [==============================] - 61s 111ms/step - loss: 0.0163 - accuracy: 0.9948 - val_loss: 0.0306 - val_accuracy: 0.9943
Epoch 50/50
545/545 [==============================] - 64s 118ms/step - loss: 0.0087 - accuracy: 0.9978 - val_loss: 0.0310 - val_accuracy: 0.9959
```

Figura 3.6. Entrenamiento de los modelos

Para ello se empleó el método `model.fit()`, el cual recibe como parámetros los generadores de imágenes de entrenamiento y validación.

- Generador de entrenamiento: alimentó al modelo con lotes de imágenes (de tamaño `batch_size`) hasta cubrir el total de muestras definidas en `steps_per_epoch`.
- Número de épocas (epochs=50): se configuró el entrenamiento para repetirse durante 50 ciclos completos sobre el conjunto de entrenamiento.
- Validación: de forma paralela, en cada época se evaluó el desempeño con el conjunto de validación, especificando los pasos correspondientes (`validation_steps`). Esto permitió monitorear métricas como accuracy (precisión) y loss (función de pérdida), (véase la Figura 3.7), tanto en entrenamiento como en validación.

```
# Evaluar el modelo en el conjunto de validación
loss, accuracy = model.evaluate(validation_generator)
print(f"Accuracy en validación: {accuracy*100:.2f}%")
```

```
78/78 ━━━━━━━━━━━━━━━━━━━━ 31s 391ms/step - accuracy: 0.9723 - loss: 0.0982
Accuracy en validación: 97.14%
```

Figura 3.7. Evaluación de los modelos

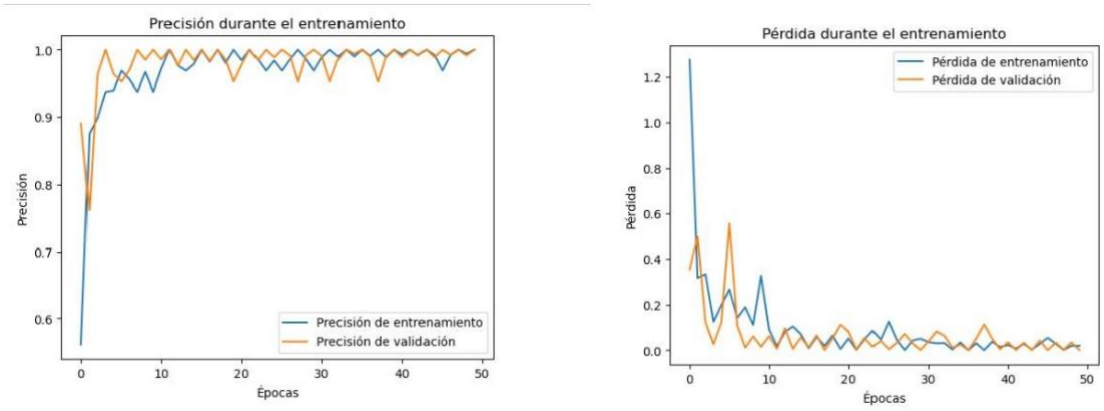


Figura 3.8. Resultados del entrenamiento y su perdida

En la Figura 3.8, en las primeras épocas se observa una mejora progresiva:

- Epoch 1: precisión de 57.81% en entrenamiento y 87.87% en validación.
- Epoch 2: precisión de 87.5% en entrenamiento y reducción de la pérdida.

Esto muestra un aprendizaje rápido de las características relevantes de las imágenes. Mientras que la pérdida (Loss) cae rápidamente a partir de la época 7.

3.6.1 Evaluación del rendimiento

Después de varias pruebas en el entrenamiento, se optimizó el modelo. Evaluado el rendimiento de clasificación la métrica Exactitud (Accuracy), véase Figura 3.9.

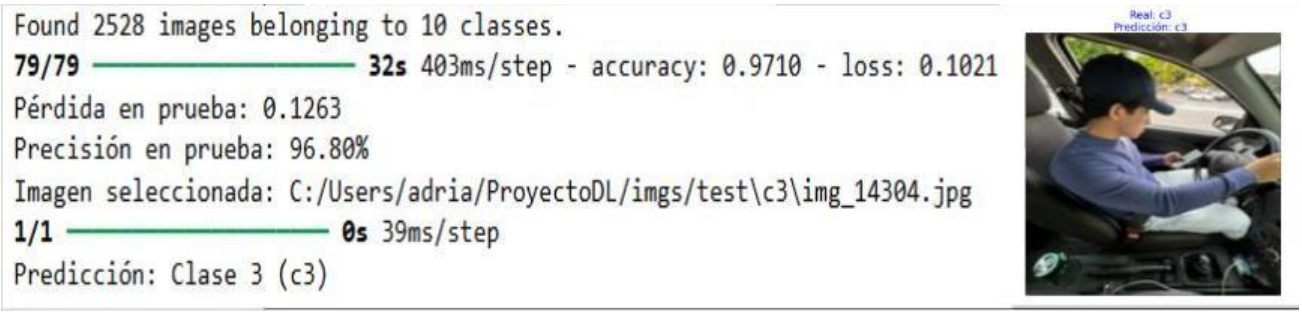


Figura 3.9. Código de la prueba del modelo

La arquitectura propuesta mostró buenos resultados, dando una exactitud del 96.80% en los mejores casos, logrando predecir la imagen mostrada en el costado derecho como C3.

3.7 Entrenamiento y evaluación del modelo CNN + Algoritmos de Clasificación

Se mantuvo la misma arquitectura base de la red neuronal convolucional (CNN); sin embargo, se realizó una modificación en la capa de salida. En lugar de utilizar la función de activación Softmax, esta capa fue eliminada y reemplazada por algoritmos de aprendizaje supervisado tradicionales, tales como K-Nearest Neighbors (KNN) y Random Forest (RF), los cuales se emplearon en la etapa final de clasificación.

Con esta modificación, la red neuronal funciona como un extractor de características en lugar de efectuar directamente la clasificación final. En específico, se conservaron únicamente las capas convolucionales y de pooling, generando vectores de características a partir de la salida de la última capa convolucional aplanada (flattened). Estos vectores fueron utilizados posteriormente como entrada para los clasificadores KNN y RF, los cuales fueron entrenados de manera independiente utilizando las etiquetas originales de las clases.

El procedimiento se desarrolló en dos etapas:

1. Extracción de características: se utilizó la CNN previamente entrenada para crear un modelo truncado que finaliza en la penúltima capa.
2. Entrenamiento de clasificadores tradicionales: los vectores de características extraídos en el paso anterior se emplearon para entrenar los modelos KNN y RF.

Estos modelos fueron implementados mediante la librería Scikit-learn, lo que permitió desacoplar las etapas de representación y decisión, facilitando la comparación del desempeño entre clasificadores tradicionales sobre un mismo espacio de características.

El enfoque híbrido busca aprovechar la capacidad de la CNN para extraer características representativas de las imágenes, combinándola con la efectividad de modelos supervisados tradicionales en la toma de decisiones. Esto permite explorar diversas estrategias para mejorar la generalización del modelo y evaluar el impacto de diferentes técnicas de clasificación en el rendimiento global del sistema.

Obteniendo los siguientes resultados:

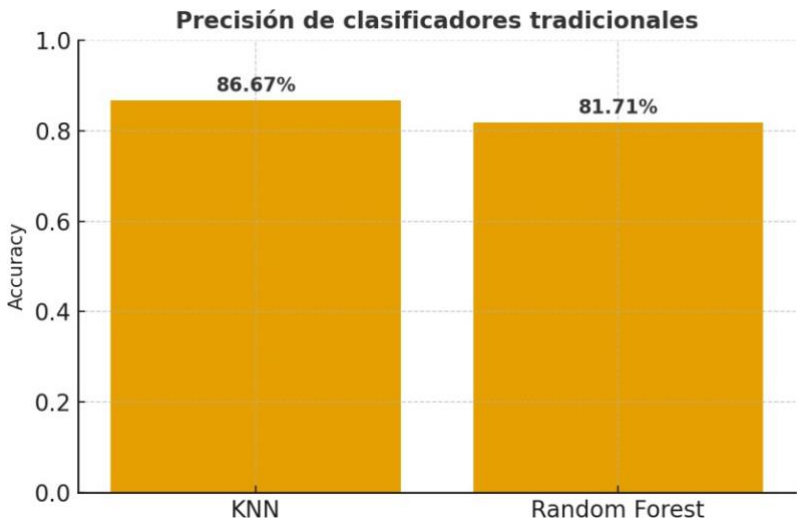


Figura 3.10. Rendimiento de algoritmos de clasificación

Se observa que el modelo KNN alcanzó una exactitud del 86.67%, superando al clasificador Random Forest, que obtuvo un 81.71%. Estos resultados evidencian que, dentro del enfoque híbrido planteado, el KNN logró un mejor aprovechamiento de las características generadas por la red neuronal convolucional para la tarea de clasificación. Sin embargo, no logró superar los resultados obtenidos con la primera arquitectura y su función de activación.

3.7.1 Matriz de confusión Random Forest

La Fig. 3.11 muestra la matriz de confusión correspondiente a los resultados, obtenidos sobre el conjunto de imágenes de prueba utilizando el algoritmo Random Forest. En la diagonal principal se observa un alto número de aciertos en la predicción de la mayoría de las clases, lo que refleja un desempeño consistente del modelo. Sin embargo, también se identifican ciertos errores de clasificación, como en la clase c3, donde 31 instancias fueron clasificadas erróneamente como c0, y en la clase c8, donde 39 instancias fueron clasificadas como c2. Estas confusiones indican que las características extraídas por la CNN presentan cierta similitud en estas clases específicas.

En general, la matriz confirma que el modelo presenta un elevado número de aciertos en la mayoría de las categorías, con algunos errores concentrados en clases particulares. Este análisis permite identificar posibles áreas de mejora, ya sea en el preprocesamiento de las imágenes o en la optimización de los hiperparámetros del clasificador.

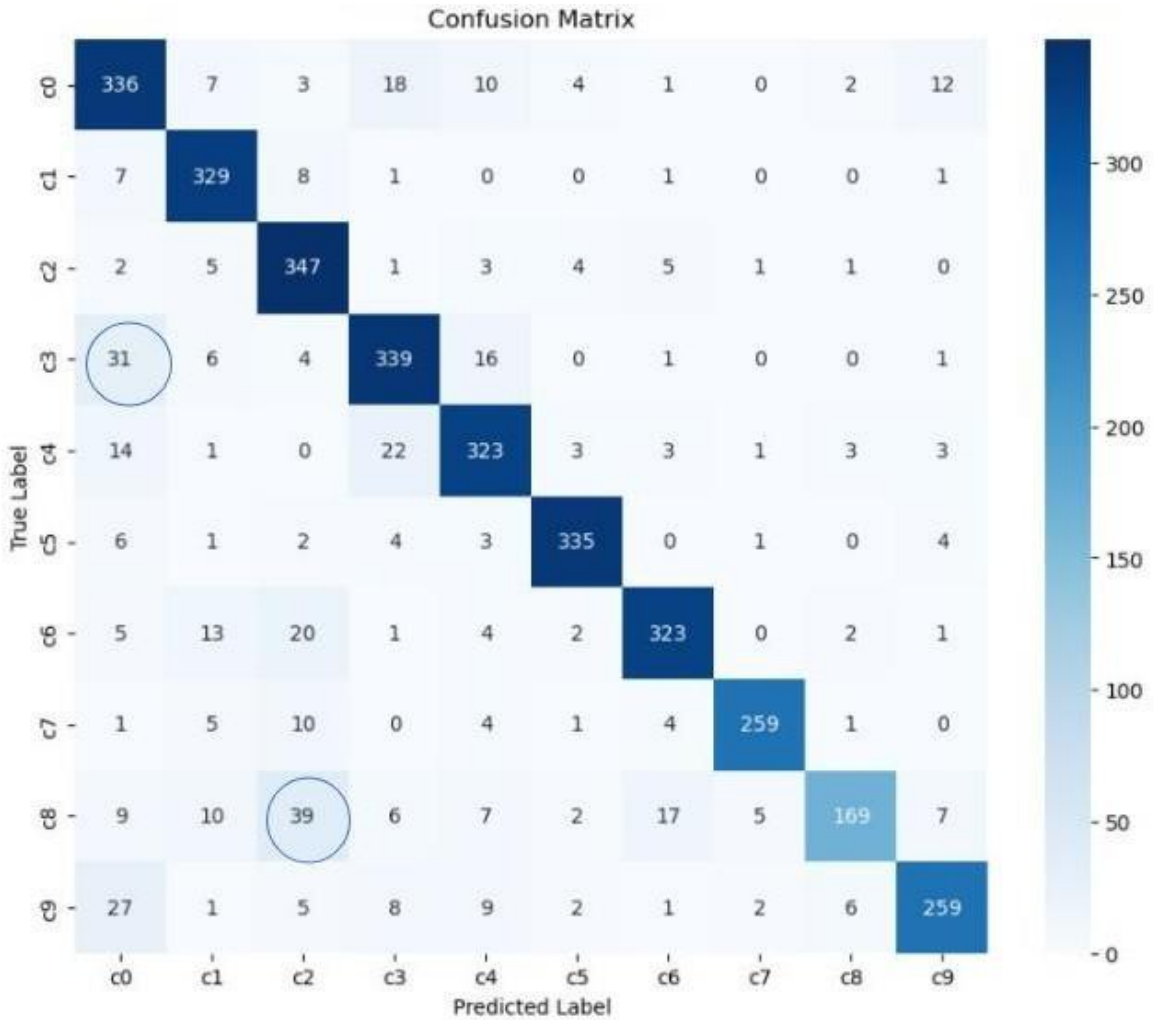


Figura 3.11. Matriz de confusión en las pruebas del modelo Random Forest

3.7.2 Matriz de confusión K-Nearest neighbors

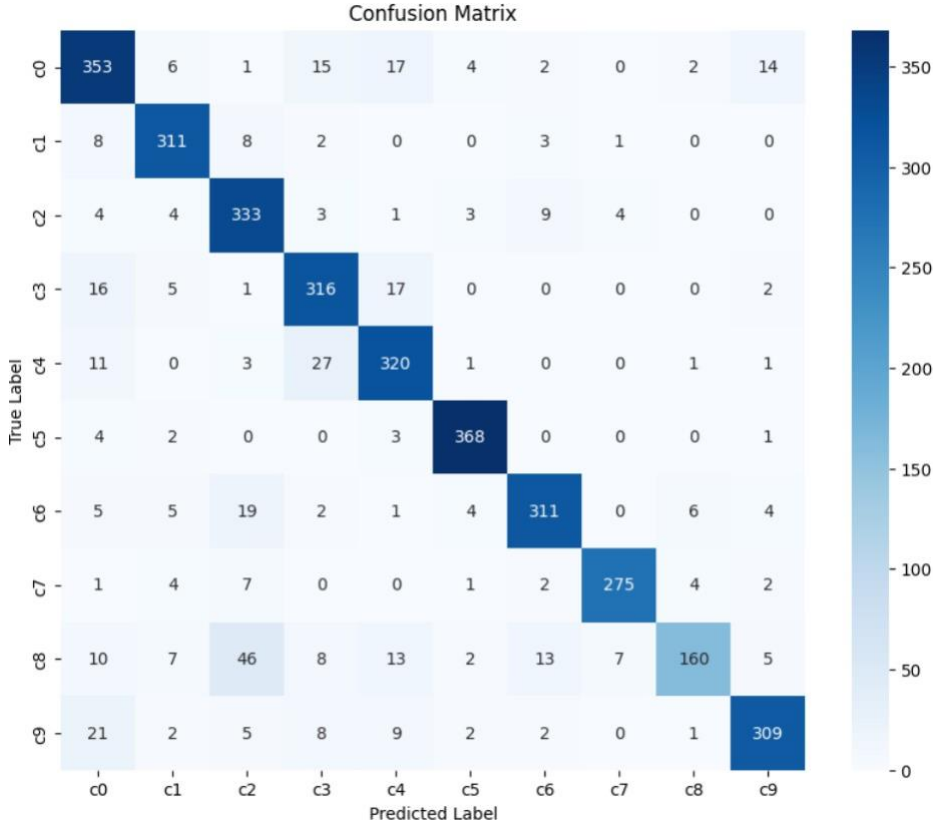


Figura 3.12. Matriz de confusión en las pruebas del modelo K-Nearest Neighbors

La Fig. 3.12 muestra la matriz de confusión obtenida con el clasificador K-Nearest Neighbors (KNN) sobre el conjunto de imágenes de prueba. En la diagonal principal se observan varios aciertos, lo que indica que el modelo logra reconocer correctamente muchas de las clases analizadas.

Sin embargo, también se presentan algunos errores de clasificación. Por ejemplo, en la clase c3 se observa que varias imágenes fueron confundidas con c0, y en la clase c8 se registran confusiones con c2 y c9. Esto puede deberse a que algunas posturas o posiciones entre clases son muy parecidas, lo que dificulta que el modelo las distinga correctamente.

En general, los resultados muestran que el modelo KNN tiene un desempeño aceptable, especialmente en clases con imágenes similares. Estos resultados ayudan a identificar qué categorías podrían mejorarse con un ajuste de parámetros o un preprocesamiento más preciso de las imágenes.

3.8 Modelo YOLOv8

Para la red neuronal YOLOv8 (véase Figura 3.13), la figura representa su arquitectura compuesta por varios bloques, en donde se combina capas de convolución, maxpooling y capas completamente conectadas. La arquitectura toma como entrada una imagen RGB, a partir de ella realiza el proceso de extracción de caracteristicas. Una vez completado el proceso, la salida pasa a la capa totalmente conectada para la identificación y realizar la clasificación. Se realizaron diferentes pruebas utilizando tanto el modelo large como el modelo nano, con el fin de evaluar el desempeño del modelo en la identificación de posturas de conducción peligrosas considerando escenarios de distinta complejidad y capacidad computacional.

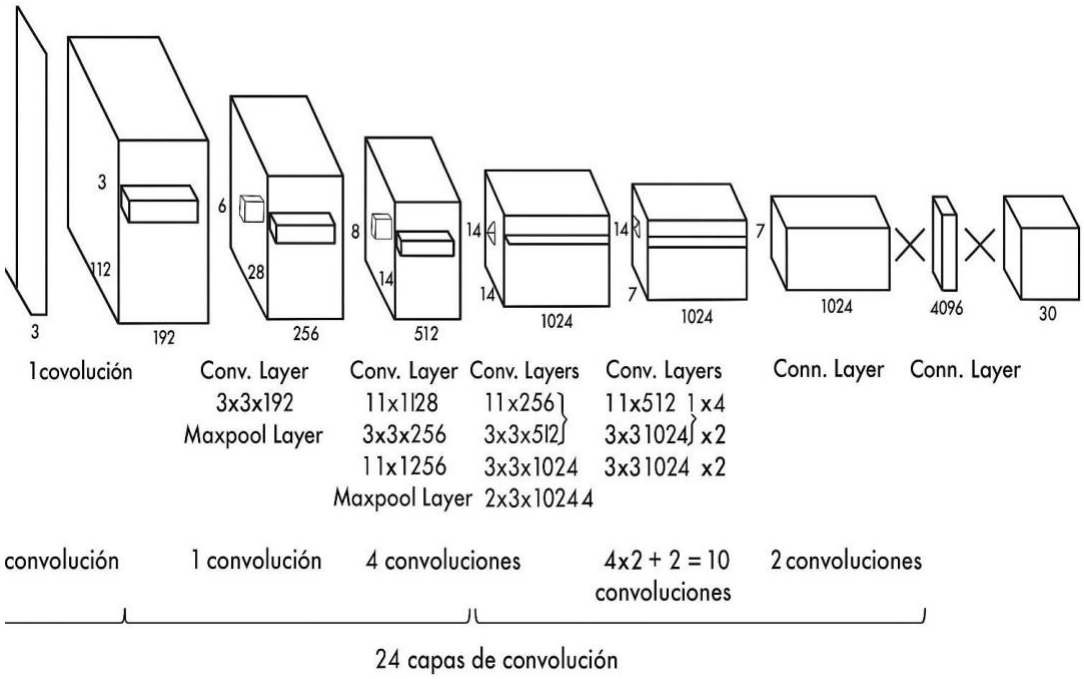


Figura 3.13. Arquitectura del Modelo YOLOv8.

Para comenzar con el uso de la red neuronal profunda YOLOv8, se llevó a cabo un preprocesamiento de las imágenes utilizando la herramienta en línea de Roboflow (véase Figura 3.14).

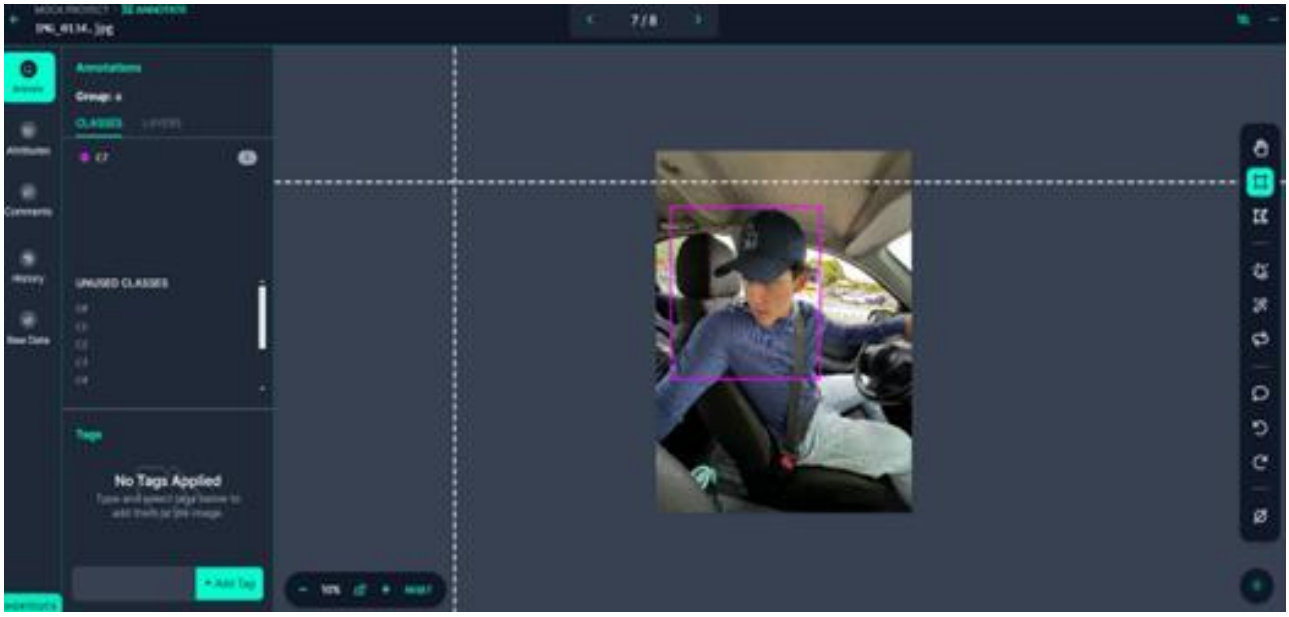


Figura 3.14. Etiqueta de posturas de conducción con RoboFlow.

Este proceso incluyó:

- Limpieza de muestras.

  Para el proceso de limpieza de las muestras, se revisó manualmente cada imagen dentro del conjunto de datos y se eliminaron imágenes duplicadas. En Roboflow se utilizaron las herramientas de inspección y vista previa para verificar la consistencia de las imágenes antes de iniciar el etiquetado.

- Etiquetado de las posturas según las diez clases.

  El etiquetado se realizó directamente en Roboflow utilizando un bounding box (cajas de marcado) para señalar la región del conductor. Las posturas fueron clasificadas siguiendo las diez clases oficiales del conjunto 'Distracted Driver Detection' (C0–C9). Cada imagen fue etiquetada manualmente. El conjunto final quedó distribuido de la siguiente forma:

  C0 – Conducción segura: 257 imágenes

  C1 – Texto con la mano derecha.: 245 imágenes

  C2 – Llamada mano derecha: 257 imágenes

  C3 – Texto con la mano izquierda: 254 imágenes

  C4 – Llamada mano izquierda: 242 imágenes

  C5 – Uso de radio o video: 257 imágenes

C6 – Bebiendo: 254 imágenes

C7 – Buscando objetos detrás: 256 imágenes

C8 – Maquillando/Peinando:257 imágenes

C9 – Conversando: 256 imágenes

⊄ Ajuste del formato y las dimensiones de las imágenes.

Todas las imágenes fueron convertidas al formato JPG. Durante el preprocesamiento en Roboflow, utilizando algunas herramientas que nos ofrece (Véase fig. 3.15) las imágenes fueron redimensionadas a 360x360 píxeles. Este tamaño ayudara a tener un mejor desempeño computacional.

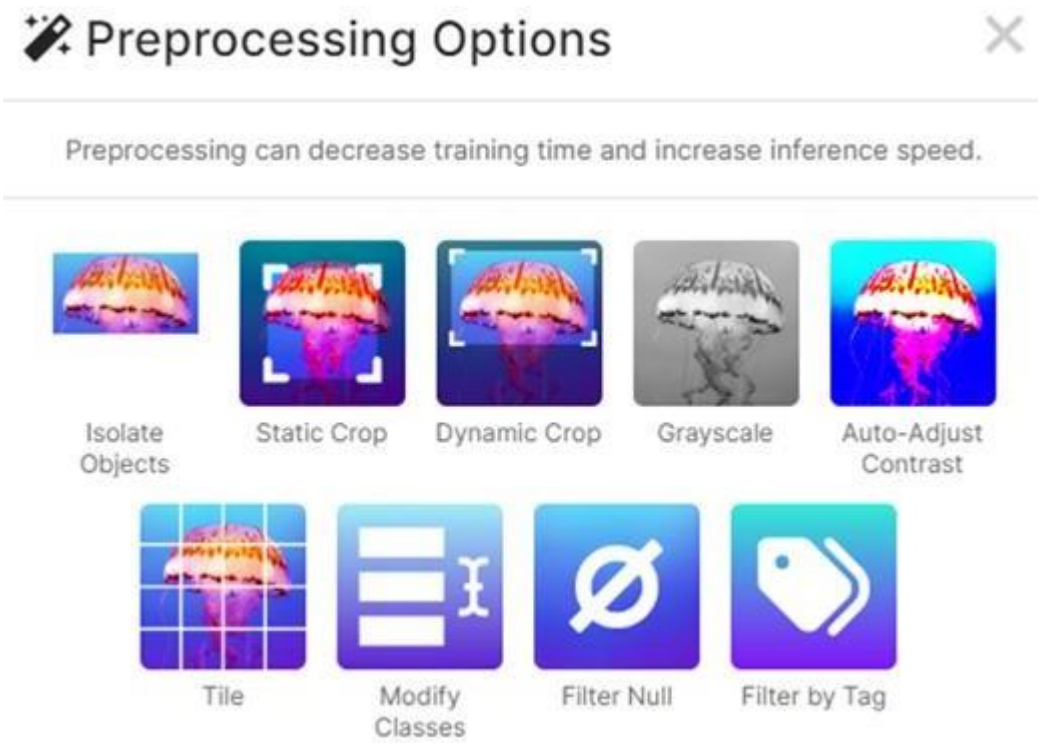


Figura 3.15. Herramientas de preprocesamiento de imágenes Roboflow.

Una vez etiquetadas las imágenes con RoboFlow, se procedió a utilizar la arquitectura de YOLOv8 nano para su entrenamiento (Figura 3.16).

```
50 epochs completed in 0.554 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 6.3MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.74 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 168 layers, 3,007,598 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [00:05<00:00,  1.53it/s]
                   all        249        249      0.899      0.938       0.95      0.745
                    c0         23         23      0.854          1      0.992      0.895
                    c1         25         25      0.975          1      0.995       0.87
                    c2         24         24      0.907          1      0.995      0.889
                    c3         26         26      0.974          1      0.995      0.886
                    c4         26         26      0.973          1      0.995      0.819
                    c5         24         24      0.972      0.958       0.96      0.742
                    c6         28         28      0.976          1      0.995        0.8
                    c7         25         25      0.988          1      0.995      0.754
                    c8         24         24      0.681      0.833      0.891      0.408
                    c9         24         24      0.689      0.583      0.685      0.388
Speed: 0.3ms preprocess, 2.9ms inference, 0.0ms loss, 4.3ms postprocess per image
Results saved to runs/detect/train2
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figura 3.16. Entrenamiento con YOLOv8 nano.

La Fig. 3.16 muestra la salida del proceso de validación del modelo YOLOv8 nano tras completar 50 épocas de entrenamiento. El modelo fue entrenado utilizando una GPU (Tesla T4, 15959MiB) con un total de 168 capas, 37,087,598 parámetros y 8 gradientes, logrando un rendimiento de:

- Precisión (P): 0.938
- Recall (R): 0.952
- mAP@50: 0.938
- mAP@50-95: 0.743

Estos valores indican un alto nivel de exactitud en la detección de objetos, especialmente en la métrica mAP@50, lo que significa que el modelo es capaz de localizar y clasificar correctamente la mayoría de las instancias en las imágenes de validación (véase Figura 3.17).

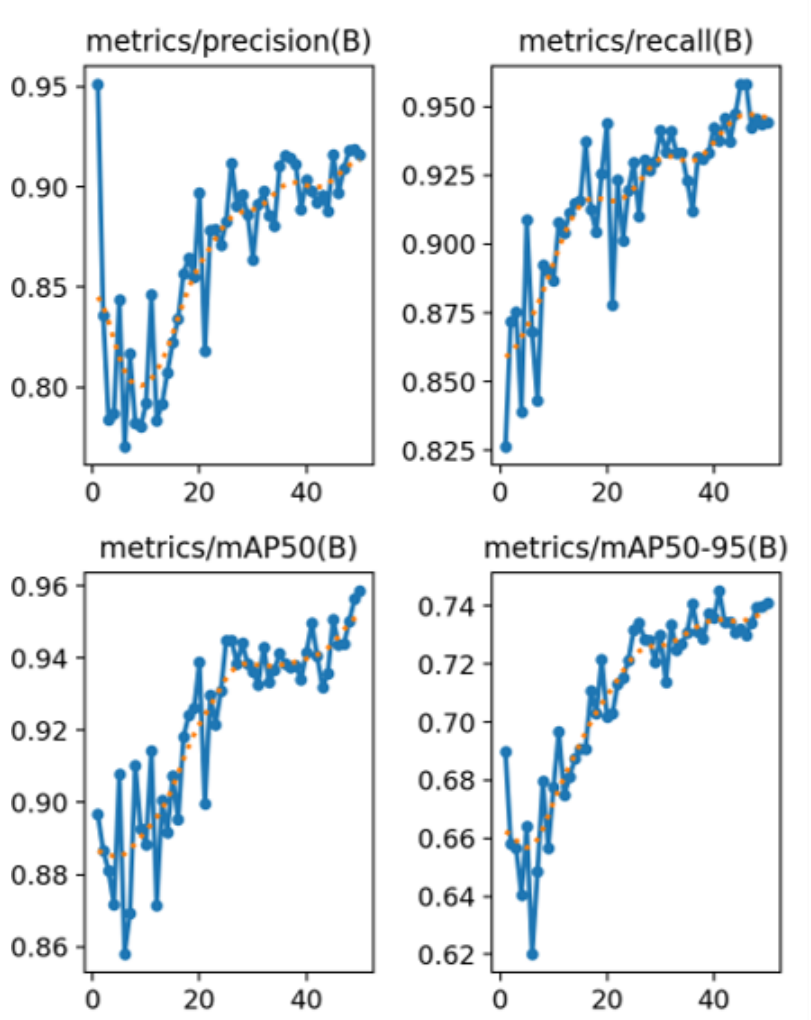


Figura 3.17. Avance de las métricas de desempeño del modelo YOLOv8 nano durante el entrenamiento.

- En la Fig. 3.17, se muestra que el modelo mejoró en cada métrica con el entrenamiento al utilizar más épocas.
- Precisión y Recall altos (~0.9) indican que el modelo es bueno detectando objetos.
- mAP@50 (~0.9) es alto, lo que indica que detecta correctamente la mayoría de los objetos con un 50% de coincidencia.
- mAP@50-95 (~0.7) es menor, lo que muestra que, a umbrales más exigentes, la precisión baja un poco.
- Se comienza con el entrenamiento de YOLOv8 en su versión nano con 100 épocas.

```
100 epochs completed in 1.061 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.3MB

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 168 layers, 3,007,598 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [00:04<00:00,  1.92it/s]
                   all        249        249      0.932      0.953      0.968      0.749
                    c0         23         23      0.874          1      0.992       0.89
                    c1         25         25      0.978          1      0.995      0.859
                    c2         24         24      0.936          1      0.995      0.899
                    c3         26         26      0.979          1      0.995      0.887
                    c4         26         26          1      0.997      0.995      0.802
                    c5         24         24      0.978      0.958      0.971      0.788
                    c6         28         28      0.971          1      0.995      0.793
                    c7         25         25      0.992          1      0.995      0.745
                    c8         24         24      0.868      0.821      0.899      0.395
                    c9         24         24      0.748       0.75      0.849      0.434
Speed: 0.3ms preprocess, 2.8ms inference, 0.0ms loss, 2.8ms postprocess per image
Results saved to runs/detect/train
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figura 3.18. Entrenamiento de la arquitectura YOLOv8 Nano.

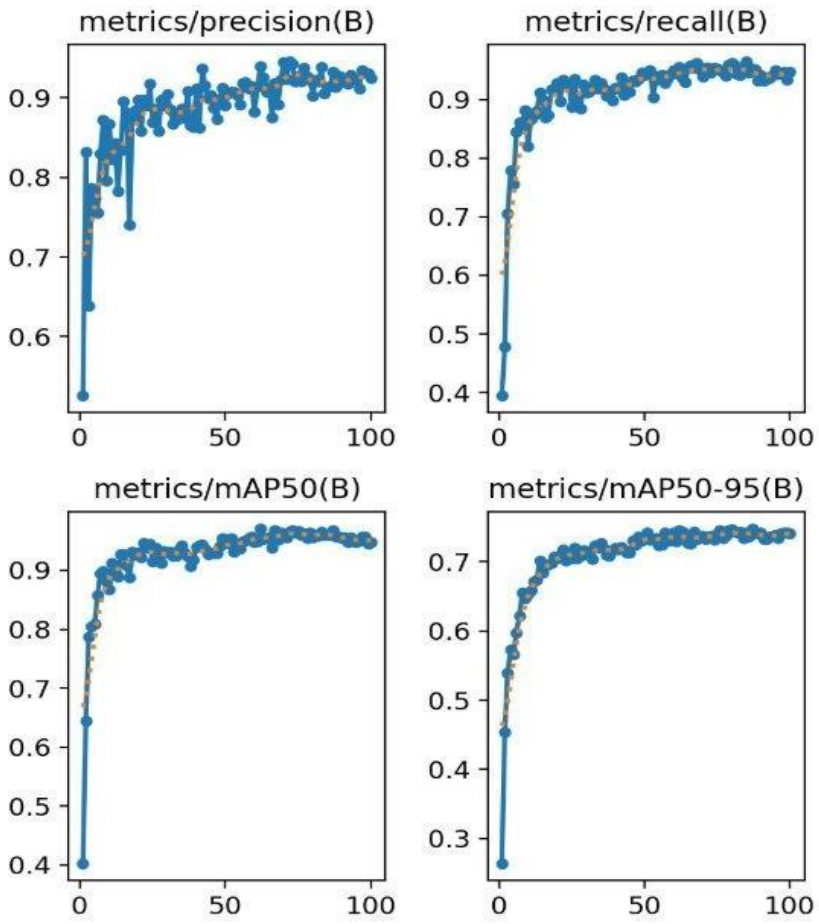


Figura 3.19 Avance de las métricas de desempeño del modelo YOLOv8 nano durante el entrenamiento con 100 épocas.

Como se aprecia en la figura 3.19 el modelo ha mejorado en cada métrica con el entrenamiento.

- Precisión y Recall altos (~0.9) indican que el modelo es bueno detectando objetos.
- mAP@50 (~0.9) es alto, lo que indica que detecta correctamente la mayoría de los objetos con un 50% de coincidencia.
- mAP@50-95 (~0.7) es menor, lo que muestra que, a umbrales más exigentes, la precisión baja un poco

En la figura 3.20 se realiza el entrenamiento del modelo YOLOv8 en su versión large.

```
50 epochs completed in 1.534 hours.
Optimizer stripped from /content/runs/detect/train/weights/last.pt, 87.7MB
Optimizer stripped from /content/runs/detect/train/weights/best.pt, 87.7MB

Validating /content/runs/detect/train/weights/best.pt...
Ultralytics 8.3.231 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 112 layers, 43,614,318 parameters, 0 gradients, 164.9 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 1.3it/s 6.2s
                   all        249        249      0.926      0.953      0.956      0.753
                    c0         23         23      0.862      0.957      0.932      0.847
                    c1         25         25      0.974          1      0.995      0.849
                    c2         24         24      0.937          1      0.995      0.899
                    c3         26         26      0.972          1      0.995      0.912
                    c4         26         26      0.975          1      0.995      0.812
                    c5         24         24      0.971      0.958      0.969      0.804
                    c6         28         28      0.874          1      0.995      0.774
                    c7         25         25          1      0.988      0.995      0.771
                    c8         24         24      0.875      0.917      0.946      0.532
                    c9         24         24      0.821      0.708      0.741      0.333
Speed: 0.2ms preprocess, 16.0ms inference, 0.0ms loss, 3.9ms postprocess per image
Results saved to /content/runs/detect/train
```

Figura 3.20. Entramiento con YOLOv8 large con 50 épocas.

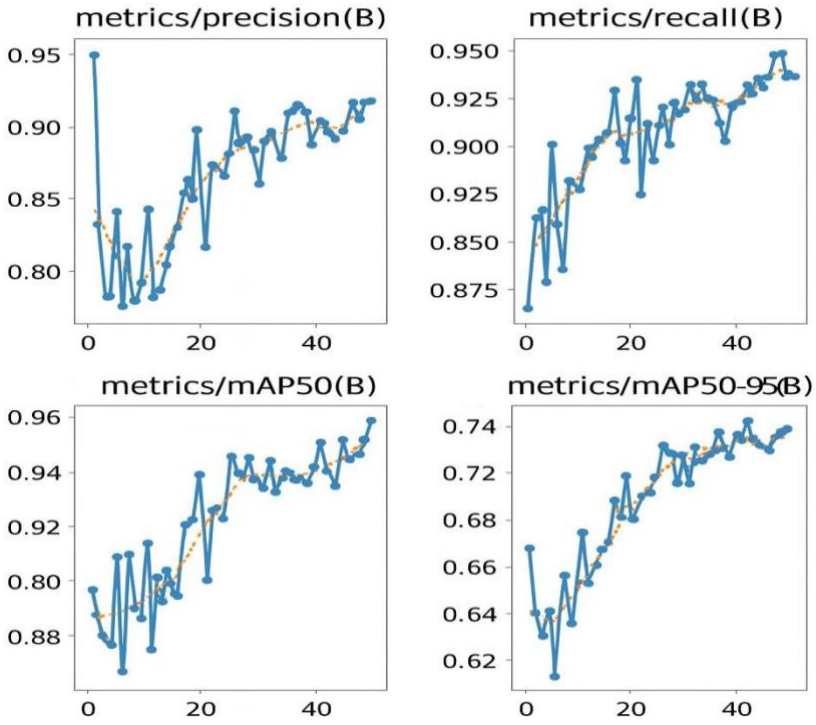


Figura 3.21. Avance de las métricas de desempeño del modelo YOLOv8 large durante el entrenamiento con 20 épocas.

Al observar los resultados de la figura 3.21, se puede ver que el modelo fue mejorando conforme avanzaron las épocas de entrenamiento. En general, tanto la precisión como el recall presentan una tendencia ascendente y se estabilizan en valores altos, lo que indica que el modelo aprende correctamente a detectar los objetos.

De igual forma, el mAP@50 alcanza valores cercanos a 0.95, lo que refleja un muy buen desempeño en la detección general. Por su parte, el mAP@50-95 también aumenta de manera constante hasta aproximadamente 0.74, mostrando que el modelo logra generalizar de forma aceptable, aunque con mayor variación en los umbrales más estrictos.

```
100 epochs completed in 3.367 hours.
Optimizer stripped from /content/runs/detect/train2/weights/last.pt, 87.7MB
Optimizer stripped from /content/runs/detect/train2/weights/best.pt, 87.7MB

Validating /content/runs/detect/train2/weights/best.pt...
Ultralytics 8.3.225 🚀 Python-3.12.12 torch-2.8.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 112 layers, 43,614,318 parameters, 0 gradients, 164.9 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ──────────── 8/8 1.3it/s 6.0s
                   all        249        249      0.918      0.946      0.957      0.752
                    c0         23         23      0.874          1      0.979      0.891
                    c1         25         25      0.976          1      0.995      0.854
                    c2         24         24      0.932          1      0.995      0.897
                    c3         26         26      0.947          1      0.995      0.896
                    c4         26         26      0.978          1      0.995      0.822
                    c5         24         24      0.974      0.958      0.984      0.775
                    c6         28         28      0.977          1      0.995      0.775
                    c7         25         25      0.978          1      0.995      0.751
                    c8         24         24      0.852      0.792      0.901      0.485
                    c9         24         24      0.692      0.708      0.734      0.379
Speed: 0.3ms preprocess, 15.6ms inference, 0.0ms loss, 3.6ms postprocess per image
Results saved to /content/runs/detect/train2
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figura 3.22. Entramiento con YOLOv8 large 100 épocas.

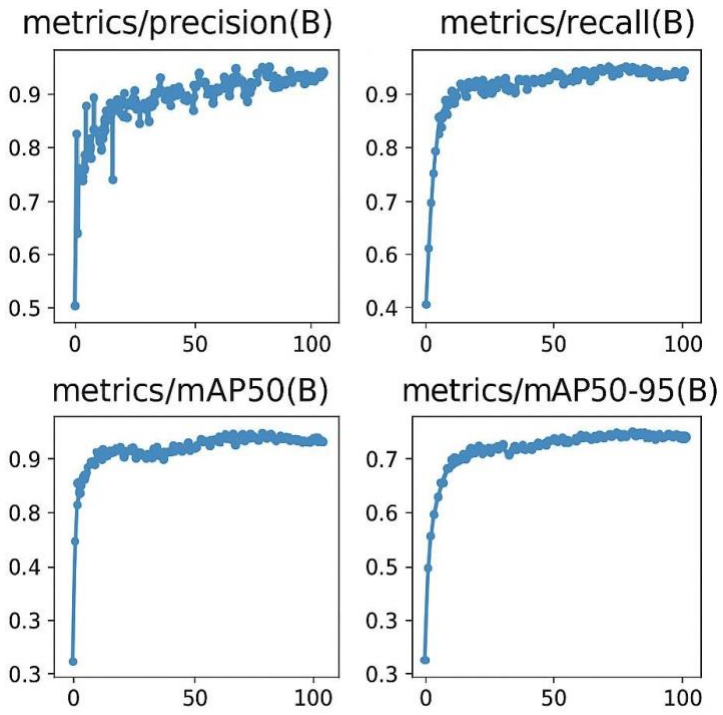


Figura 3.23. Avance de las métricas de desempeño del modelo YOLOv8 large durante el entrenamiento con 100 épocas.

El análisis clase por clase evidencia que varias categorías alcanzaron métricas cercanas al 100% (véase figura 3.23.):

- c1, c2, c3, c4, c5 y c6: presentan valores de precisión y recall muy altos (≥ 0.99), lo que indica un desempeño sobresaliente en la identificación de estas clases.
- c0 y c7: muestran también resultados consistentes, con mAP@50 por arriba de 0.74.
- c8 y c9: destacan como las clases con menor rendimiento. En particular, c9 obtuvo un mAP@50-95 de 0.388, indicando dificultad del modelo para generalizar en esta categoría.

El modelo presenta un excelente rendimiento global, con valores de precisión y recall que superan el 93%. Sin embargo, la disminución de desempeño en clases (c8 y c9) sugiere la necesidad de:

- Aumentar la cantidad de instancias disponibles en el conjunto de entrenamiento para dichas clases.
- Ajustar hiperparámetros para mejorar la capacidad de generalización en las categorías c8 y c9.

3.9 Arquitectura CNN + DepthMap

Para este experimento, se optó por emplear la misma arquitectura CNN propuesta, pero para este caso se incorporó un proceso adicional de preprocesamiento de imágenes denominado DepthMap. Este procedimiento consiste en la generación de un mapa de profundidad a partir de las imágenes originales, con el objetivo de resaltar no únicamente las características bidimensionales tradicionales (como bordes, texturas y formas), sino también información relacionada con la percepción de profundidad y la geometría de los objetos presentes en la escena.

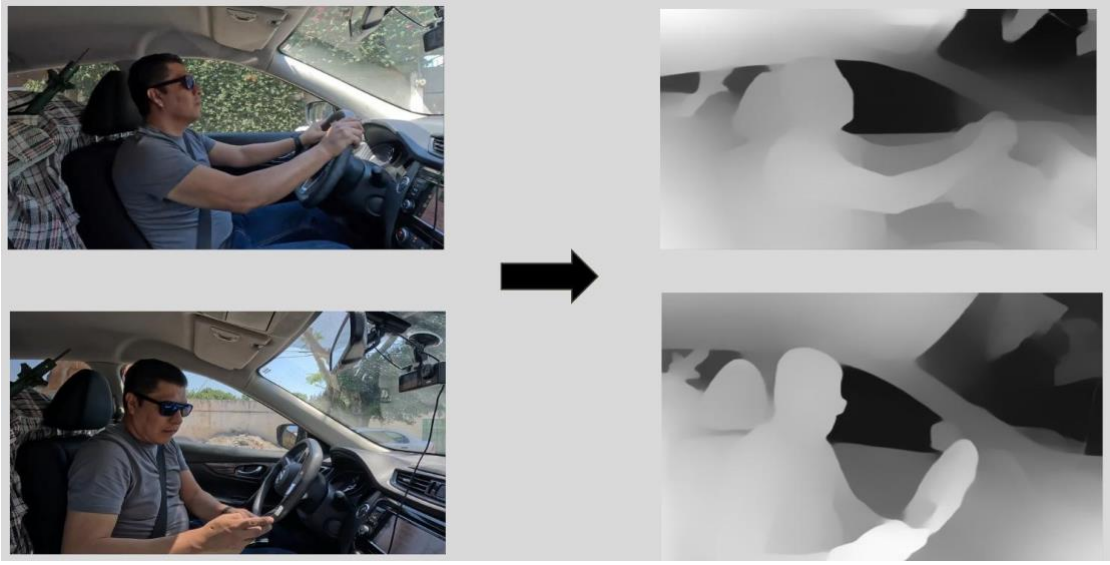


Figura 3.24. Transformación de una imagen original a DepthMap

El preprocesamiento mediante DepthMap transforma cada imagen en una representación en escala de grises, donde los valores de intensidad corresponden a la distancia relativa de cada punto respecto a la cámara. De esta manera, las regiones más cercanas aparecen en tonos claros, mientras que las más alejadas se representan en tonos oscuros. Esta representación aporta información estructural adicional que puede enriquecer la etapa de extracción de características en las capas convolucionales, favoreciendo que el modelo detecte patrones espaciales (Ranftl, Bochkovskiy & Koltun, 2021).

Para este experimento se utilizó la misma arquitectura de 9 capas, incluyendo sus 4 capas convolucionales con MaxPooling, la capa de aplanamiento y las capas densas finales, además de las etapas arriba descritas de preprocesamiento, se utilizó adicionalmente DepthMap, con el fin de mejorar la exactitud y la capacidad de generalización del modelo en comparación con las imágenes originales. Esta estrategia representa un enfoque híbrido que combina la potencia de las CNNs con técnicas avanzadas de visión por computadora.

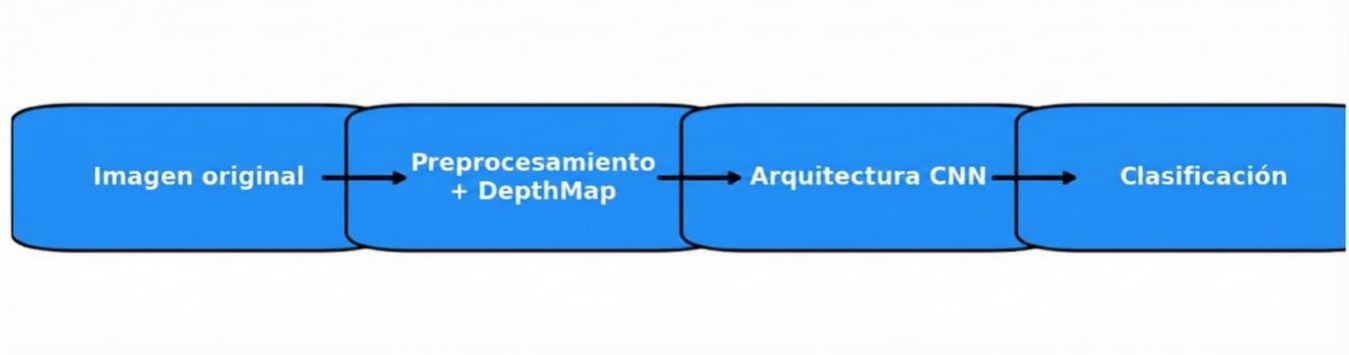


Figura 3.25. Flujo del preprocesamiento + DepthMap y arquitectura CNN.

Con el procedimiento mostrado en la figura 3.25, se realizó la prueba para este modelo. Comenzando con la imagen original con un formato de 1487 x 807 pixeles como se hace referencia en la figura 3.3, de esta manera, se realizó el ajuste de dimensiones a 150 x 150 pixeles. Adhiriendo posteriormente en el preprocesamiento a DepthMap. El modelo fue entrenado con un total de 50 épocas (Véase figura 3.26.) .

```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size)
```

```
Epoch 1/10
1090/1090 [==============================] - 281s 258ms/step - loss: 0.0718 - accuracy: 0.9777 - val_loss: 0.0823 - val_accuracy: 0.9762
Epoch 2/10
1090/1090 [==============================] - 115s 106ms/step - loss: 0.0617 - accuracy: 0.9802 - val_loss: 0.0790 - val_accuracy: 0.9798
Epoch 3/10
1090/1090 [==============================] - 119s 109ms/step - loss: 0.0495 - accuracy: 0.9836 - val_loss: 0.1167 - val_accuracy: 0.9744
Epoch 4/10
1090/1090 [==============================] - 114s 104ms/step - loss: 0.0500 - accuracy: 0.9840 - val_loss: 0.0745 - val_accuracy: 0.9863
```

Figura 3.26. Entrenamiento del modelo CNN + DepthMap.

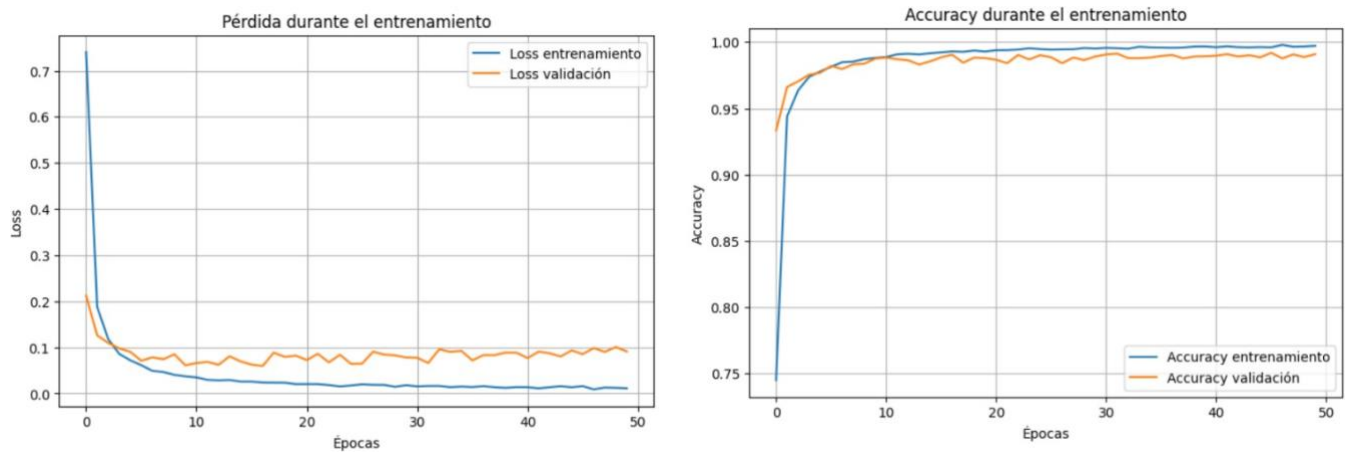


Figura 3.27 Resultados con DepthMap en la pérdida y el accuracy en la etapa de entrenamiento.

La figura 3.27 muestra cómo fue aprendiendo el modelo a lo largo de las 50 épocas de entrenamiento. En la primera gráfica, se observa que la pérdida del modelo con DepthMap disminuye rápidamente en las primeras épocas y luego se estabiliza.

En la segunda gráfica de la misma figura, se aprecia que el modelo alcanza un nivel de precisión muy alto desde las primeras épocas, superando el 95% tanto en entrenamiento como en validación. Conforme avanzan las épocas, la precisión mejora ligeramente y se mantiene estable. En la figura 3.28 se muestra el resultado del modelo en su etapa de prueba.

```
1091/1091 ━━━━━━━━━━━━━━━━━━━━ 166s 152ms/step - accuracy: 0.9847 - loss: 0.0486
Pérdida en prueba: 0.0450
Precisión en prueba: 98.55%
Imagen seleccionada: C:/Users/adria/OneDrive/Escritorio/MOCA/ProyectoDL_depth/train\c7\img_37325_flip.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
Predicción: Clase 7 (c7)
```

Figura 3.28. Evaluación del desempeño de la red neuronal convolucional.

Se obtuvieron muy buenos resultados en la etapa de prueba, con las métricas siguientes: accuracy 0.9847, loss 0.0486 durante el entrenamiento y los valores de la prueba fueron del 98.55%, logrando predicciones bastantes buenas como se muestra en la figura 3.29.

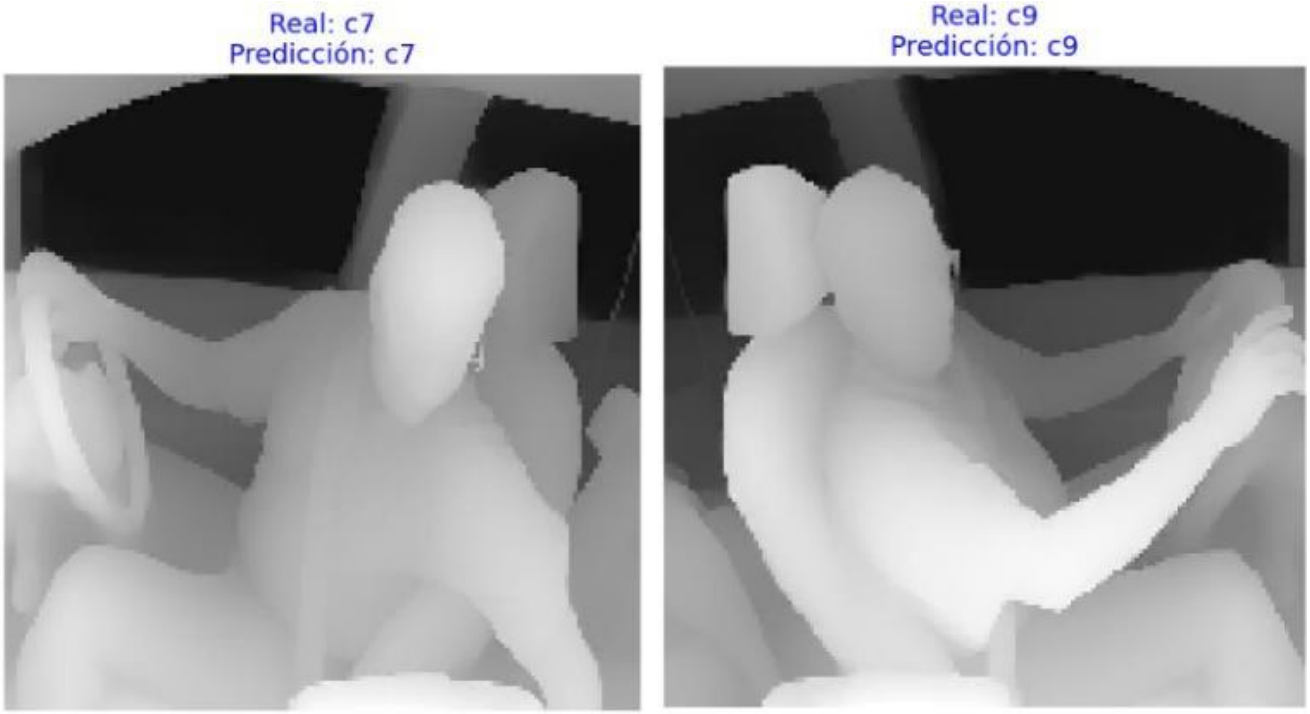


Figura 3.29. Predicción de postura de conducción con DepthMap.

Las imágenes de la figura 3.29 muestran los resultados de los dos mapas de profundidad en diferentes posturas de conducción. En ambos casos se observa al conductor, donde los tonos claros representan las partes más cercanas a la cámara y los oscuros las más lejanas. En la primera imagen, el modelo clasificó correctamente la postura c7, y en la segunda también logró identificar la clase c9. Estos resultados muestran que usando DepthMap, el modelo logró reconocer de manera adecuada las actividades del conductor.

La contribución principal es el diseño e implementación de una red neuronal convolucional propia, integrada con mapas de profundidad (DepthMap) para el reconocimiento de posturas de conducción. El uso de DepthMap, en lugar de imágenes RGB convencionales, aporta robustez ante variaciones de iluminación y reduce la información irrelevante para la tarea de clasificación. La arquitectura CNN fue diseñada con criterios de ligereza computacional, lo que permite su despliegue en dispositivos con recursos limitados (edge devices). Adicionalmente, se llevó a cabo un estudio de ablación para identificar la configuración óptima de la red, evaluando el impacto de distintos componentes sobre el desempeño final del modelo. Los resultados obtenidos, con una precisión del 98.55%, validan la efectividad de la arquitectura propuesta.

CAPÍTULO 4. RESULTADOS Y DISCUSIÓN

4.1 Comparación de resultados de los Modelos

Los resultados obtenidos muestran un análisis comparativo entre diferentes arquitecturas de clasificación en posturas de conducción.

- CNN + DepthMap: alcanzó una exactitud del 98.55%, representando el mejor desempeño general entre los modelos evaluados. El preprocesamiento con mapas de profundidad permitió a la red capturar información espacial adicional, lo cual se tradujo en una mejora significativa en la discriminación de clases.
- YOLOv8: obtuvo una precisión del 96.40%, destacando por su alta eficiencia en detección con sus bounding box. Este modelo, si bien no superó al esquema basado en DepthMap, demostró ser altamente competitivo para aplicaciones donde la velocidad de procesamiento es un factor crítico.
- CNN estándar: registró una exactitud del 96.80%, lo que evidencia un buen rendimiento en la clasificación directa sin preprocesamiento adicional. Sin embargo, su desempeño fue ligeramente inferior al del modelo combinado con DepthMap.
- CNN + KNN: presentó una exactitud del 86.67%, lo cual refleja una reducción en la capacidad de generalización al sustituir la capa de salida por un clasificador KNN. Aunque logró un rendimiento aceptable, no alcanzó la precisión de las arquitecturas con clasificación directa.
- CNN + Random Forest: fue el enfoque con menor rendimiento, obteniendo una exactitud del 81.71%. Esto sugiere que la combinación con Random Forest no resulta adecuada para explotar de manera óptima las características extraídas por la CNN.

Principales Confusiones

El análisis de errores mostró que las confusiones más frecuentes ocurrieron entre las clases:

- C8 y C2,
- C3 y C0,
- C9 y C0.

Estas confusiones reflejan similitudes en las posturas de conducción en las categorías, por lo cual ocurren esos errores en la identificación.

4.2 Estudio de ablación CNN estándar

Si bien los resultados obtenidos evidencian un alto desempeño de la arquitectura propuesta, es importante analizar en qué medida cada uno de los parámetros contribuye a diferentes resultados. Para ello, se llevó a cabo un estudio de ablación. Utilizando 12 arquitecturas de modelos seleccionados para evaluar el impacto de componentes clave de la arquitectura del modelo. Cada modelo (A1–A12) se construyó variando el número y disposición de capas convolucionales (C) y capas de MaxPooling (M), así como el número de filtros, la función de activación y el tamaño de la capa densa final.

Tabla 4.1 Estudio de ablación con CNN propuesta.

Modelo	Arquitectura (Conv/Max)	Filtros	Activación	Parámetros (Tamaño)	Accuracy
A1	1C + 1M / 1C + 1M	32, 64	ReLU + 512 Neuronas	42M (162 MB)	97.55%
A2	1C / 1C / 1C / 1C + 1M	32, 64, 128, 128	ReLU + 512 Neuronas	330M (1.23 GB)	97.38%
A3	1C / 1C / 1C + 1M	32, 64, 128	ReLU + 512 Neuronas	339M (1.27 GB)	95.53%
A4	1C + 1M / 1C / 1C	32, 64, 128	ReLU + 512 Neuronas	321M (1.20 GB)	94.70%
A5	1C / 1C / 1C / 1C + 1M	32, 64, 128, 128	ELU + 512 Neuronas	330M (1.23 GB)	88.81%
A6	1C / 1C + 1M / 1C / 1C + 1M	32, 64, 128, 128	ReLU + 512 Neuronas	76M (290 MB)	96.63%
A7	1C + 1M / 1C + 1M / 1C + 1M	32, 64, 128,	ReLU + 512 Neuronas	19M (72 MB)	97.07%
A8	1C / 1C / 1C / 1C + 1M	32, 64, 128, 128	ReLU + 256 Neuronas	165M (631 MB)	96.40%

A9	1C + 1M x3	32, 64, 128	ReLU + 256 Neuronas	9.5M (36 MB)	96.24%
A10	**1C + 1M x3**	**32, 64, 64**	**ReLU + 256 Neuronas**	**4.7M (18 MB)**	**98.10%**
A11	1C + 1M / 1C + 1M	32, 64	ReLU + 512 Neuronas	42M (162 MB)	93.75%
A12	1C + 1M	32	ReLU + 512 Neuronas	89M (341 MB)	95.61%

Modelos A1–A4: muestran arquitecturas progresivas, aumentando la profundidad y el número de filtros. Lograron precisiones entre 97.38% y 95.53%, lo que evidencia que un mayor número de parámetros no siempre garantiza un mejor desempeño (por ejemplo, A3, con 339M de parámetros, solo alcanzó 95.53%).

Modelo A5: utilizó la activación ELU en lugar de ReLU, obteniendo un desempeño inferior (88.81%), lo que sugiere que ReLU es más adecuada para este problema específico.

Modelos A6–A8: se redujo el número de parámetros de manera significativa (entre 76M y 165M) manteniendo activación ReLU. El mejor desempeño en este grupo lo obtuvo A7 con 97.07%.

Modelos A9 y A10: simplificaron aún más la arquitectura (solo tres bloques de C+M). Destaca el modelo A10, que redujo el número de parámetros a solo 4.7M (18 MB), alcanzando una precisión de 98.10%, la más alta de todos los experimentos. Esto evidencia que una arquitectura más compacta puede ser más eficiente y precisa que modelos más complejos.

Modelos A11 y A12: con arquitecturas más reducidas, obtuvieron resultados intermedios (93.75% y 95.61%, respectivamente), confirmando que una simplificación excesiva afecta la capacidad de generalización del modelo.

En el estudio podemos observar que la complejidad no siempre es sinónimo de mejor rendimiento. Aunque arquitecturas con cientos de millones de parámetros obtuvieron resultados competitivos, la configuración A10 (1C + 1M ×3, 32–64–64 filtros, ReLU + 256 neuronas densas) alcanzó la mayor precisión (98.10%), utilizando

significativamente menos recursos computacionales. Estos resultados muestran la importancia de la optimización a través de estudios de ablación para lograr modelos más ligeros, eficientes y precisos.

De acuerdo con el estudio de ablación se determinó la mejor arquitectura de red neuronal profunda para esta problemática. Los resultados de este análisis se ven presentando en la Figura 4.1.

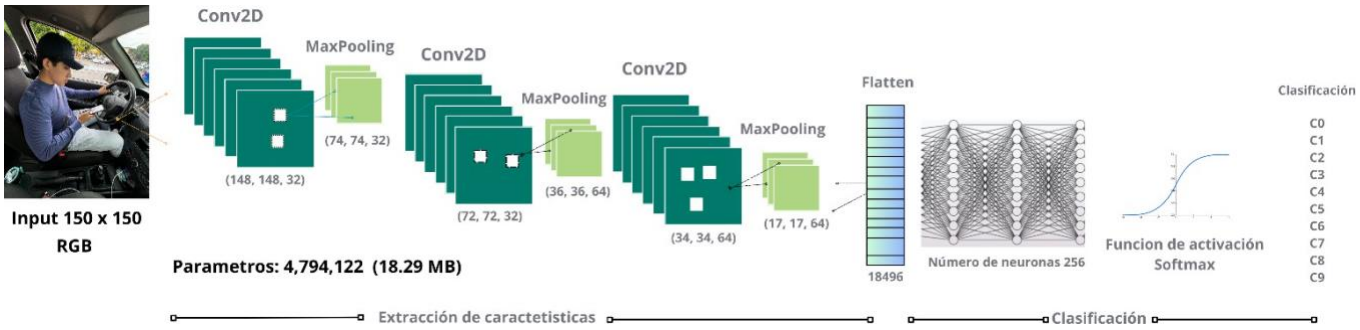


Figura 4.1. Arquitectura CNN después del estudio de ablación.

A continuación, en la Tabla 4.2, se muestra el estudio de ablación aplicado a la CNN utilizando Depth map en el preprocesamiento.

Tabla 4.2 Estudio de ablación con CNN propuesta y Depth map aplicado en el preprocesamiento.

Modelo	Arquitectura (Conv/Max)	Filtros	Activación	Parámetros (Tamaño)	Accuracy
DP1	1C + 1M / 1C + 1M	32, 64	ReLU + 512 Neuronas	32M (162 MB)	97.36%
DP2	1C + 1M / 1C + 1M / 1C + 1M	32, 64, 128,	ReLU + 512 Neuronas	19M (72 MB)	96.30%
DP3	1C + 1M x3	32, 64, 128	ReLU + 256 Neuronas	9.5M (36 MB)	97.65%
DP4	**1C+1M x 4**	**32,64,128,128**	**ReLU +512**	**3.4M(14MB)**	**98.55%**
DP5	1C + 1M x3	32, 64, 64	ReLU + 256 Neuronas	4.7M (18 MB)	97.55%

DP6	1C + 1M / 1C + 1M	32, 64	ReLU + 512 Neuronas	42M (162 MB)	97.90%
DP7	1C + 1M	32	ReLU + 512 Neuronas	89M (341 MB)	93.99%

Modelos DP1–DP3: presentan arquitecturas con varias capas convolucionales y diferentes combinaciones de filtros. Estos modelos logran precisiones entre 96.30% y 97.65%, con un número de parámetros que va desde 9.5M hasta 32M. Se observa que al aumentar filtros y neuronas no siempre se obtiene una mejora significativa en la precisión.

Modelo DP4: es el modelo que destaca claramente en esta etapa. Utiliza una arquitectura más profunda (1C + 1M ×4), con filtros 32, 64, 128 y 128, activación ReLU + 512 neuronas, pero con solo 3.4M de parámetros (14 MB). A pesar de ser el más compacto, alcanza la mejor precisión de todos: 98.55%.

Modelo DP5: mantiene una estructura similar, pero con menos filtros en las capas profundas y una densa de 256 neuronas. Su tamaño sube ligeramente a 4.7M de parámetros, logrando una precisión de 97.55%, lo que indica un desempeño competitivo por debajo de DP4.

Modelo DP6: incrementa considerablemente el número de parámetros (42M, 162 MB), usando nuevamente ReLU + 512 neuronas, logrando una precisión de 97.90%. No supera el rendimiento del modelo DP4, lo que demuestra que más parámetros no garantizan mejores resultados.

Modelo DP7: es el más pesado de toda la tabla, con 89M de parámetros (341 MB) y una arquitectura más simple en filtros, alcanzando una precisión de 93.99%, la más baja de la tabla. Este resultado deja claro que un modelo grande si no está bien balanceado, puede tener un mal desempeño.

4.3 Resultados comparativos

Techniques	DataBase	Accuracy
CNN + BiLSTM (Jiama, 2020)	A.U.C. D.D.D.	92.70%
CNN + Classification Algorithm (Al-Doori, 2021)	-	-
KNN	State Farm D.D.D.	**98.10%**
SVM	State Farm D.D.D.	95.80%
RF	State Farm D.D.D.	88.10%
HRNet + ResNet / ResNet + ResNet101 (Koay, 2021)	A.U.C. D.D.D.	94.28%
Hybrid CNN ResNet50-InceptionV3-Xception (Huang, 2020)	State Farm D.D.D.	96.74%
ResNet50 + VGG16 (Aljasim y Kashef, 2022)	State Farm D.D.D.	92.00%
VGG16 (Loyala, 2024)	State Farm D.D.D.	**98.74%**
CNN (This Research)	State Farm D.D.D.	**98.10%**
CNN + Classification algorithm (This Research)	-	-
KNN	State Farm D.D.D.	86.67%
RF	State Farm D.D.D.	81.71%
YOLOv8 (This Research)	State Farm D.D.D.	96.40%
CNN + Depthmap (This Research)	State Farm D.D.D.	**98.55%**

Figura 4.2. Resultados comparativos de esta investigación.

La tabla 4.2 presenta un análisis comparativo entre diferentes enfoques propuestos en la literatura y los modelos desarrollados en esta investigación, todos aplicados a la base de datos State Farm Distracted Driver Detection (D.D.D.), o a bases relacionadas de la Universidad Americana de El Cairo (AUC) D.D.D.

Modelos de trabajos previos

- CNN + BiLSTM (Jiama, 2020): alcanzó una precisión de 92.70% en la base de datos de la Universidad Americana de El Cairo (AUC) D.D.D., demostrando la utilidad de arquitecturas híbridas, aunque con resultados limitados frente a otros modelos más recientes.
- CNN + Classification Algorithm (Al-Doori, 2021): se reporta con State Farm como base de datos, mostrando mejoras respecto a CNN tradicional.
- KNN: logró un 98.10%, lo que evidencia su efectividad en este tipo de tareas cuando se aplican técnicas de extracción de características adecuadas.
- SVM: obtuvo una exactitud de 95.80%, confirmando su solidez como clasificador, aunque no supera a las CNN.

- RF: presentó 88.10%, el valor más bajo entre los métodos tradicionales.
- HRNet + ResNet / ResNet + ResNet101 (Koay, 2021): reportó 94.28% en la base AUC D.D.D., mostrando mejoras frente a modelos clásicos.
- Hybrid CNN ResNet50-InceptionV3-Xception (Huang, 2020): alcanzó 96.74%, confirmando el potencial de arquitecturas híbridas profundas.
- ResNet50 + VGG16 (Aljasim y Kashef, 2022): reportó 92.00%, por debajo de otros enfoques.
- VGG16 (Loyala, 2024): presentó un resultado sobresaliente con 98.74%, uno de los más altos de la tabla.

Modelos de esta investigación

- CNN estándar: alcanzó una precisión de 98.10%, mostrando un desempeño competitivo frente a arquitecturas más complejas.
- CNN + Classification Algorithm: al integrar clasificadores tradicionales (KNN y RF) en lugar de la capa de salida Softmax, los resultados fueron más bajos (86.67% con KNN y 81.71% con RF), confirmando que estos clasificadores no aprovechan al máximo las características extraídas por la CNN.
- YOLOv8: alcanzó una precisión de 96.40%, lo que lo posiciona como un modelo sólido especialmente para detección en tiempo real, aunque ligeramente por debajo de las CNN puras en clasificación directa.
- CNN + DepthMap: logró 98.55%, ubicándose entre los mejores resultados reportados y demostrando la eficacia del preprocesamiento con mapas de profundidad para enriquecer la representación de las imágenes y mejorar la capacidad de generalización.

Los resultados muestran que el modelo CNN + DepthMap propuesto en esta investigación se encuentra dentro de los enfoques más precisos (98.55%), muy cercano al rendimiento máximo reportado (98.74% con VGG16). Esto valida la pertinencia del preprocesamiento con mapas de profundidad como una estrategia competitiva frente a arquitecturas profundas de referencia en el área.

CAPÍTULO 5. CONCLUSIONES Y TRABAJO FUTURO

La presente investigación discutió y abordó un problema central: la identificación de posturas de conducción a través de visión por computadora y aprendizaje profundo. Los resultados obtenidos permiten extraer conclusiones relevantes:

1. Eficiencia de las Redes Neuronales Convolucionales (CNN):

   Las arquitecturas basadas en CNN demostraron ser altamente efectivas para la identificación y clasificación de actividades de conducción, alcanzando una exactitud del 96.80% con la arquitectura estándar. Este desempeño valida la hipótesis de que las redes profundas son apropiadas para distinguir entre actividades de conducción seguras y peligrosas.
2. Limitaciones de los clasificadores tradicionales:

   Clasificadores como KNN y Random Forest (RF), en combinación con la CNN como etapa final de decisión, mostraron un rendimiento inferior (86.67% y 81.71%, respectivamente). Estos resultados evidencian que, aunque dichos algoritmos pueden complementar el proceso de clasificación, resultan menos efectivos que las CNN. Esto pude haberse debido a la falta de una mejor sintonización de hiperparametros para KNN y RF.
3. Contribución del preprocesamiento con DepthMap:

   La integración de la técnica DepthMap como etapa de preprocesamiento representó una mejora significativa en el rendimiento del modelo, alcanzando una exactitud del 98.55%, la más alta obtenida en esta investigación. Este resultado demuestra que la incorporación de información de profundidad en las imágenes mejora la capacidad de generalización del modelo, situándolo al nivel de arquitecturas de referencia como VGG16, pero con un diseño más ligero y eficiente.
4. Aplicación práctica y relevancia social:

   La identificación precisa de actividades de conducción peligrosas podría implementarse en sistemas de monitoreo inteligentes dentro de unidades de transporte público o privado, contribuyendo directamente a la reducción de accidentes y al mejoramiento de la seguridad de pasajeros y conductores. Este enfoque refuerza la importancia de aplicar tecnologías de visión artificial y aprendizaje profundo en contextos reales, especialmente en el ámbito de la movilidad y el transporte seguro.
5. Impacto académico y proyección futura:

Los resultados alcanzados no solo confirman la efectividad de las CNN, sino que también abren la puerta a nuevas líneas de investigación enfocadas en arquitecturas híbridas y en el uso de técnicas de preprocesamiento avanzado. La efectividad del enfoque con DepthMap demuestra que la técnica podría incrementar aún más los resultados en los sistemas de clasificación de posturas de conducción. Al finalizar este trabajo, se logró cumplir el objetivo principal: implementar un modelo basado en aprendizaje profundo capaz de reconocer y clasificar diferentes actividades de conducción. A lo largo del proceso se utilizaron repositorios existentes y también se recopilaron imágenes propias, lo que permitió entrenar, probar y ajustar los modelos.

El preprocesamiento, la limpieza de datos y la clasificación por clases ayudaron a estructurar mejor la metodología, y gracias a ello fue posible comparar diferentes modelos hasta identificar la red más adecuada para esta problemática. Los resultados mostraron que el modelo logró distinguir entre posturas normales de conducción y actividades distractoras. Con base en los resultados obtenidos, se cumple la hipótesis inicial: Las redes profundas (CNN) y la incorporación de mapas de profundidad (DepthMap) en el preprocesamiento mejora significativamente el desempeño de clasificación de actividades de conducción.

Los resultados confirman que el uso de visión artificial y aprendizaje profundo son una herramienta viable y efectiva para el monitoreo del comportamiento del conductor, lo cual abre la puerta a futuras mejoras e implementación en dispositivos como Jetson o Rasbpberry. Como trabajo futuro se propone la implementación del modelo en dispositivos Jetson Nano, de manera que el modelo se monte en el dispositivo y funcione en tiempo real para la identificación de posturas peligrosas al momento de conducir.

REFERENCIAS

Agüero, D., Almeida, G., Espitia, M., Flores, A., & Espig, H. (2014). Uso del teléfono celular como distractor en la conducción de automóviles. *Salus, 18*(2), 27–34. https://ve.scielo.org/scielo.php?pid=S1316-71382014000200006&script=sci_arttext

Al-Doori, S. K. S., Taspinar, Y. S., & Koklu, M. (2021). Distracted driving detection with machine learning methods by CNN based feature extraction. *International Journal of Applied Mathematics Electronics and Computers, 9*(4), 116–121. https://doi.org/10.18100/ijamec.1035749

Aljasim, M., & Kashef, R. (2022). *Driver distraction detection using ResNet50 and VGG16* [Preprint]. Kaggle. https://www.kaggle.com/competitions/state-farm-distracted-driver-detection

Aljure Jiménez, Y. (2021). *Clasificación de las flores con redes neuronales convolucionales* [Trabajo de grado, Universidad de Antioquia]. Repositorio Institucional UdeA. https://bibliotecadigital.udea.edu.co/bitstream/10495/24683/1/AljureYalila_2021_ClasificacionImagenesFlores.pdf

Cifuentes, L., Valenzuela, M., Morales, K., Carrasco, M., & Besoain-Saldaña, A. (2022). *Condiciones de seguridad, salud, trabajo y empleo de trabajadores y trabajadoras de plataformas digitales (Reparto y transporte de pasajeros)* (1.ª ed.). Instituto de Seguridad Laboral. https://www.isl.gob.cl/wp-content/uploads/Estudios-Condiciones-de-Seguridad-Salud-Trabajo-y-Empleo-en-Trabajadores-trabajadoras-de-plataformas-digitales.pdf

Huang, C., Wang, X., Cao, J., Wang, S., & Zhang, Y. (2020). HCF: A hybrid CNN framework for behavior detection of distracted drivers. *IEEE Access, 8*, 109335–109349. https://doi.org/10.1109/ACCESS.2020.3001159

IBM. (2023). *¿Qué es un algoritmo k-NN?* https://www.ibm.com/mx-es/topics/knn

IBM. (s. f.). *What is a neural network?* IBM Think. Recuperado el 16 de septiembre de 2025, de https://www.ibm.com/think/topics/neural-networks

Ilemobayo, J. A., Durodola, O. I., Alade, O., Awotunde, O. J., Adewumi, T. O., Falana, O., Ogungbire, A., Osinuga, A., Ogunbiyi, D., Ifeanyi, A., Odezuligbo, I. E., & Edu, O. E. (2024). Hyperparameter tuning in machine learning: A comprehensive review. *Journal of Engineering Research and Reports, 26*(6), 388–395. https://doi.org/10.9734/jerr/2024/v26i61188

Instituto Nacional de Estadística y Geografía (INEGI). (2023a). *Estadística de accidentes de tránsito terrestre en zonas urbanas y suburbanas – Año 2023*. https://www.inegi.org.mx/app/buscador/default.html?q=atus

Instituto Nacional de Estadística y Geografía (INEGI). (2023b). *Vehículos de motor registrados en circulación 2023. Información anual*. https://www.inegi.org.mx/rnm/index.php/catalog/1019/related-materials

Instituto Nacional de Estadística y Geografía (INEGI). (2023c). *Accidentes de tránsito terrestre en zonas urbanas y suburbanas (ATUS)*.

https://www.inegi.org.mx/programas/accidentes/default.html

Instituto Mexicano del Transporte. (2025). *Aspectos básicos aplicables a la evaluación de la maniobrabilidad de vehículos de carretera* https://imt.mx/resumen-boletines.html?IdArticulo=375&IdBoletin=141

Jiménez Alfaro, A. D., & Díaz Ospina, J. V. (2022). Revisión sistemática de literatura: Técnicas de aprendizaje automático (Machine Learning). *Cuaderno Activa, 13*(1), 113–121. https://doi.org/10.23850/25004115.849

Kaggle. (2016). *State Farm Distracted Driver Detection*. https://www.kaggle.com/competitions/state-farm-distracted-driver-detection

Koay, H. V., Chuah, J. H., Chow, C.-O., Chang, Y.-L., & Rudrusamy, B. (2021). Optimally-weighted image-pose approach (OWIPA) for distracted driver detection and classification. *Sensors, 21*(14), Artículo 4837. https://doi.org/10.3390/s21144837

Landa-Buendía, A., Hernández-Aguilar, J. A., Ortiz-Hernández, J., Díaz-González, L., & Oubram, O. (2025). Identification of dangerous driving patterns through computer vision and deep learning. En *Artificial Intelligence – COMIA 2025: 17th Mexican Congress (Proceedings, Part II)* (pp. 3–14). Springer. https://doi.org/10.1007/978-3-031-97910-1_1

Loyola Ayque, M. (2024). *Reconocimiento de comportamiento de conducción distraída utilizando aprendizaje por currículo en una red profunda preentrenada VGG16* [Tesis de grado, Universidad Católica San Pablo]. Repositorio Institucional UCSP. https://hdl.handle.net/20.500.12590/18115

Olabe, X. B. (1998). *Redes neuronales artificiales y sus aplicaciones*. Publicaciones de la Escuela de Ingenieros.

Ranftl, R., Bochkovskiy, A., & Koltun, V. (2021). Vision transformers for dense prediction. En *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 12179–12188). https://doi.org/10.48550/arXiv.1907.01341

Raschka, S. (2018). *STAT 479: Machine learning lecture notes*. University of Wisconsin–Madison. https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf

Xing, Y., Lv, C., Wang, H., Cao, D., Velenis, E., & Wang, F. Y. (2019). Driver activity recognition for intelligent vehicles: A deep learning approach. *IEEE Transactions on Vehicular Technology, 68*(6), 5379–5390. https://doi.org/10.1109/TVT.2019.2908425

Yan, C., Coenen, F., & Zhang, B. (2016). Driving posture recognition by convolutional neural networks. *IET Computer Vision, 10*(2), 103–114. https://doi.org/10.1049/iet-cvi.2015.0175

REPOSITORIO GITHUB

https://github.com/AdrianLandaBue/MOCA-TESIS

Cuernavaca, Morelos a 13 de mayo del 2026.

MTRO. ELMER IVAN SÁNCHEZ RABADAN
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisora de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante **Adrian Landa Buendia**, con matrícula 10025160, con el título **Identificación de Comportamientos de Conducción Peligrosos en Servicios de Transporte Mediante aprendizaje profundo.***,* por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dra. Lorena Díaz González
Profesora- investigadora
Centro de Investigación en Ciencias

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

LORENA DIAZ GONZALEZ | Fecha:2026-05-13 10:13:27 | FIRMANTE

ssgI9iPziM70rBM6HdwneZDdLzjGfBI98kmOt9MvXoJgt2W9qoEMHv4YZAqnapXkpzMTen3632fwcbDuFdzYjTiXru2LfcJH8Kkz3NapK57M+aIYZttfqX6RGxz9GFrJLMLz7xSOLQ2
g0aNvvL5GHqrgkMT7fxV/aJWbAHC8qXY6J9tV8QBxidNtYOkmNHSRe3wqnvWPRujvylWmRdIshF5CrmASH8p+LFc+TwooPS/LuHWPNViFvs4neYLFTANw5Msy8ITLy2ZHL9oll
22usqoQfavS4PFQX/GxaoNkLkuNWIbsNejzGrRmE5jM4rHboVFsRHl+diCYVr+1RvZAIQ==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

bTgQKzOP0

https://efirma.uaem.mx/noRepudio/DV7ywN4L4p9MEoP1axzQAsl78FdrYYFe

Cuernavaca, Morelos a 14 de abril del 2026.

MTRO. ELMER IVAN SANCHEZ RABADAN
ENCARGADO DE DESPACHO DE LA DIRECCION DE LA FACULTAD
DE CONTADURIA, ADMINISTRACION E INFORMATICA DE LA UAEM.
PRESENTE.

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Adrian Landa Buendia, con matrícula 10025160, con el título **Identificación de Comportamientos de Conducción Peligrosos en Servicios de Transporte Mediante aprendizaje profundo.**, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. Outmane Oubram
Profesor- investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

OUTMANE OUBRAM | Fecha:2026-04-14 19:31:38 | FIRMANTE

PQ6ddrOc1jzlhl2/tp/SPXGhgiCOdQ0kG/AWcD0u7HWaOeep96AZ51e4KrJlYy9nQcrG32NWRA23C3DQ1ejA/6mVs2owLX1XrFe0jAdyIR6YbGzELns/w7tKLUgREVs366lJxFOsa
Xke/gghoxN0AXkfuy0Ja1lYmHeuwTS+xtaUej14qa1QUEfe8HCdayEIx3Ur/EENWr2+XbOteGHWSMU+OM3Mu8bcp0N/S5hThLPKlfTrmyBNsfYZlIEtWASAs7hljKO4lTlpK0cXyCK
fgZjJ6WS8SrcqJWdOxvKzVgAUXzMyb0B/d+xpsZoJ7ISKabCKHRBYrZUbTei+vlhbYg==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

O7yqza3rv

https://efirma.uaem.mx/noRepudio/WWu87BN1XBTapgtj36LuxyshEX6GqPLj

Cuernavaca, Morelos a 30 de marzo del 2026.

Mtro. Elmer Iván Sánchez Rabadán
DIRECTOR INTERINO DE LA FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Adrian Landa Buendia, con matrícula 10025160, con el título **Identificación de Comportamientos de Conducción Peligrosos en Servicios de Transporte Mediante aprendizaje profundo.**, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. Martín Gerardo Martínez Rangel
Profesor- investigador
Facultad de Contaduría, Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

MARTIN GERARDO MARTINEZ RANGEL | Fecha:2026-04-02 12:13:07 | FIRMANTE

MeWrbK1gPRzQCBF3dEFCnjyPJYxiL4iQPlVknBpKEOb/a9Su+RHVUQqJC6HQT7klcrWysYp7MOk1qI0Fl3RqgPlMQKioX4mohuGK85N/xiUZUfVV0kQUl4lOUkleouiAMa+c1BU
PzFZPs63nAKzqfICB3S0BGcvroyYRBRqILx0MK7DJFvc7xRsOSSzRL1tlcSuMbKUp0juNiA9yqQKeY39P/VxhXIQyFxTh3OPPOOiWfQH2wVV+kFTZj4/YlQus219CtB6yCNWG1m
aBrEltBbBkDsc1lhCEyygazLM6UX1PBqC+5m53rzbXE6P+y50A7L02o7eEU3IB5VLwPXak/w==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

y7ulBmtfG

https://efirma.uaem.mx/noRepudio/YF68yB4Wd4gMRsqL50LnQgKZIIXIn14k

Cuernavaca, Morelos a 11 de mayo del 2026.

MTRO. ELMER IVAN SÁNCHEZ RABADAN
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Adrian Landa Buendia, con matrícula 10025160, con el título **Identificación de Comportamientos de Conducción Peligrosos en Servicios de Transporte Mediante aprendizaje profundo.**, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. Javier Ortiz Hernández
Profesor- investigador
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Morelos a 19 de mayo del 2026.

MTRO. ELMER IVAN SÁNCHEZ RABADAN
ENCARGADO DE DESPACHO DE LA DIRECCIÓN DE LA FACULTAD
DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA DE LA UAEM.
PRESENTE

En mi carácter de revisor de tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la Maestría en Optimización y Cómputo Aplicado, del estudiante Adrian Landa Buendia, con matrícula 10025160, con el título **Identificación de Comportamientos de Conducción Peligrosos en Servicios de Transporte Mediante aprendizaje profundo.**, por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que la Universidad Autónoma del Estado de Morelos tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta

Dr. José Alberto Hernández Aguilar
Profesor- investigador
Facultad de Contaduría Administración e Informática

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento con firma electrónica UAEM, soportada por el certificado vigente a la fecha de su elaboración y con efectos plenos de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS PUBLICADOS en el ÓRGANÓ INFORMATIVO UNIVERSITARIO "ADOLFO MENÉNDEZ SAMARÁ" número 117 de fecha 20 de abril de 2021.

Sello electrónico

JOSE ALBERTO HERNANDEZ AGUILAR | Fecha:2026-05-19 11:17:22 | FIRMANTE

BZ/MoIA4movrVydJp5YGoRqyc3JvS9tz8XCxi633vXGXAM4LILoUocwTihb9O2hu6zqobK4tltS9Kmmz3h1FWnyjq1J/pNWbm8jxUs7CSDmNq71TgoTZ7B1yTA48E0ItZKtAwzhMU
tHtTsxB2dGRYXzQEdw5Cs0Eo3HgMJMlEefGdx7m/74ml06mpR0UCIP6IIHiaz4RMnqvzzaIwRYFAWI913u0p3+zTQo/pz/dNN98XxGRZSh1Q80nM1WqeXMN6Z5Et3xsXUZx5ws
Xx/Atbgqv8kobPbSNXXhm9ktUQ38HH0Pb7onGkJsSF2irrqpRpxaKl9vrn1F4QWvwEZraew==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

BL9W8SVCu

https://efirma.uaem.mx/noRepudio/2vXYmdvZKGzSjmtNSBfVqu7EUv21oBnr