UNIVERSIDAD VERACRUZANA

INSTITUTO DE INVESTIGACIONES EN
INTELIGENCIA ARTIFICIAL

Reducción de parámetros de una red neuronal convolucional para estimación de ángulos en imágenes de conducción autónoma mediante neuroevolución

T E S I S

Como requisito para obtener el grado de
Maestría en Inteligencia Artificial

P R E S E N T A

José David Velazco Muñoz

Director de Tesis:
Dr. Héctor Gabriel Acosta Mesa

Co-Director de Tesis:
Dr. Efrén Mezura Montes

Xalapa Enríquez, Veracruz
Septiembre 2024

"La verdadera inteligencia reside en la capacidad de aprender y evolucionar, adaptándose a cada nuevo desafío. El futuro pertenece a quienes transforman la complejidad en simplicidad, desafiando los límites de la tecnología para crear soluciones que aún no imaginamos."

ChatGPT, OpenAI, 2024.

Agradecimientos

Quiero agradecir a Dios por siempre haber estado presente en estos dos años de maestría, por haber cumplido un sueño y poder estudiar en la Universidad Veracruzana.

Agradezco a mi madre, por sus consejos, recomendaciones y por estar presente en todo momento, brindándome su apoyo incondicional y su amor inquebrantable. Ella ha sido una fuente constante de sabiduría y fortaleza, y su ejemplo ha sido una guía invaluable en mi vida.

Agradezco a mi padre, por su templanza y su generosidad en todo momento. Su serenidad y comprensión han sido un pilar fundamental para mantenerme firme y enfocado en mis objetivos. Su apoyo constante y su ejemplo de integridad me han inspirado a dar lo mejor de mí en cada paso de este camino.

De igual forma, agradezco a mi hermana, Dulce Velazco, por siempre apoyarme emocionalmente y estar presente en las necesidades que se fueran presentando. Su comprensión y disposición para ayudar en cualquier circunstancia han sido un soporte invaluable.

También quiero agradecer a mis amigos 'legendarios', quienes fueron pieza clave en mi desarrollo personal y académico. Su motivación constante y su compañía me dieron el impulso necesario para seguir adelante en la maestría, especialmente en los momentos más desafiantes. Su apoyo y camaradería han sido esenciales para mi éxito.

Quiero expresar mi sincero agradecimiento al Dr. Héctor Acosta por su apoyo y confianza en este proyecto, incluso en los momentos de duda y los desafíos que enfrentamos. Agradezco profundamente sus enseñanzas, valiosos consejos y recomendaciones que han sido pilares fundamentales para el desarrollo de esta tesis. Reconozco también la importancia de sus correcciones y advertencias, las cuales han sido guías indispensables en esta investigación. Su compromiso y dedicación han sido verdaderamente importantes, y estoy profundamente agradecido por haber contado con su guía y respaldo.

Quiero agradecer al Dr. Efrén Mezura, quien ha sido un profesor excepcional y un ejemplo de pasión por la ciencia. Su dedicación, conocimiento y compromiso han dejado una huella imborrable en mi formación académica y profesional. Gracias por su valiosa enseñanza y por inspirarme a seguir explorando y aprendiendo en el fascinante mundo de la inteligencia artificial.

Quiero expresar mi profundo agradecimiento a mi generación de la MIA 2022-2024 por su incondicional apoyo durante clases, sus explicaciones claras y la ayuda en los diversos procesos que se presentaron a lo largo del camino. Gracias a Leo Flores, Jaime Rangel y en su momento Otoniel Buenrostro. Muy especialmente, quiero destacar a Raúl García, un amigo incondicional que encontré en este recorrido. Su amistad y apoyo han sido fundamentales para superar los desafíos de esta etapa. Gracias, Raúl, por tantas risas, el ánimo constante y la motivación a lo largo de estos años de maestría.

Agradezco a mis padrinos, toda la generación de estudiantes del DIA del grupo de investigación de C O V N N E C – A p p que estuvieron durante mi trayectoria en la MIA. Por compartir siempre su conocimiento, experiencias y valiosas recomendaciones, han sido un pilar fundamental en mi desarrollo académico y profesional. Quiero expresar mi gratitud especialmente a Clemente Hernández, David Herrera, Carlos López, José Fuentes, Arnulfo Barradas así como también a José Luis Morales.

Agradezco al Dr. Mario Castelán por haberme abierto las puertas del CINVESTAV-Saltillo, por compartir generosamente su vasto conocimiento y por su constante presencia. Mi gratitud también se extiende a todo su grupo de investigación por su apoyo y colaboración invaluables.

De igual forma, agradezco al Dr. Sergio Henández, Mtra. Erandi Barrientos y Dr.Mario Castelán, por a ver aceptado ser mis sinodales y tomarse el tiempo de revisar esta tesis.

Agradezco el apoyo del Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT) a través de una beca con el número CVU 1220680 para llevar a cabo estudios de posgrado en la Universidad Veracruzana.

Resumen

Los vehículos autónomos representan el futuro de la movilidad, donde la seguridad y eficiencia en la navegación son primordiales. Para lograr estos objetivos, dependen de sistemas avanzados de percepción, procesamiento y toma de decisiones que les permitan interpretar su entorno y actuar en consecuencia. Estos sistemas están equipados con múltiples sensores, cámaras, radares y LIDAR que capturan grandes cantidades de datos en tiempo real. Estos datos son procesados por complejos algoritmos de inteligencia artificial que, en última instancia, deben tomar decisiones críticas como frenar, acelerar o girar el volante, todo ello con la máxima precisión y en fracciones de segundo.

Dentro de estos sistemas de toma de decisiones, la estimación precisa del ángulo de dirección es uno de los factores clave. Este parámetro es esencial porque determina la orientación del volante, lo que a su vez influye directamente en la trayectoria del vehículo. En un entorno de conducción autónoma, donde las situaciones pueden cambiar rápidamente.

A medida que los vehículos autónomos enfrentan entornos más complejos, la optimización de los modelos de predicción de ángulo de dirección se vuelve una prioridad. La capacidad de ajustar este ángulo, lleva a la necesidad de innovar en el diseño y entrenamiento de las redes neuronales que procesan estos datos.

En nuestra búsqueda de optimizar las redes neuronales convolucionales para predecir ángulos de dirección en vehículos autónomos, fue implementada la neuroevolución, una fusión innovadora de redes neuronales y algoritmos evolutivos. Cada individuo en la población se representa mediante genes que codifican tanto la arquitectura como los pesos de la red neuronal, facilitando así el proceso de aprendizaje y optimización. Este enfoque no solo busca minimizar la cantidad de parámetros en los modelos, sino que también se enfoca en reducir la latencia y mejorar el tiempo de respuesta, aspectos cruciales en aplicaciones de conducción autónoma.

En la fase inicial, los individuos (redes neuronales convolucionales) se entrenan utilizando 1000 imágenes etiquetadas con ángulos de dirección en entornos vehiculares. El proceso de entrenamiento se realiza durante 30 épocas, con lotes de 10 imágenes. Además, se ha afinado el algoritmo genético, ajustando la tasa de mutación al 0.7 % y la tasa de cruce al 0.9 %, valores determinados mediante una búsqueda en rejilla (grid search).

La red neuronal propuesta con 35,972 parámetros, resultado de este proceso de neuroevolución, fue comparada con modelos reconocidos en la literatura, como PilotNet (233,701 parámetros), RestNet50 (25,636,699 parámetros), VGG16 (138,357,456 parámetros), VGG16-LSTM (156,223,443 parámetros) y una CNN optimizada mediante PSO (1,335,758 parámetros) . Los procesos de entrenamiento fueron estandarizados para garantizar una evaluación justa, utilizando 10,000 imágenes, 50 épocas y una tasa de aprendizaje del 0.0001.

Los resultados no solo revelan diferencias significativas en el número de parámetros entre las RNC evaluadas, sino que también muestran que la red propuesta logra un desempeño comparable con modelos mucho más complejos. Este hallazgo sugiere que la red propuesta no solo es más eficiente en cantidad de parámetros, sino que también ofrece una latencia y un tiempo de respuesta más bajos, factores cruciales para su aplicación en entornos de conducción autónoma en tiempo real. Este enfoque permite descubrir arquitecturas de redes neuronales compactas, eficientes y con bajas latencias para la predicción del ángulo de dirección en vehículos autónomos.

Índice

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

Los vehículos autónomos han estado en el pensamiento del ser humano desde la creación del primer vehículo terrestre, esta idea fue creciendo conforme iba avanzando la tecnología, industria e investigación. Decir autónomo a un vehículo es entender que tiene las capacidades del ser humano al estar conduciendo, o sea, que pueda percibir el entorno donde se está desplazando. Estos vehículos representan una innovación significativa en el transporte y tienen potencial para la movilidad y logística en el planeta.

Hablar de vehículos autónomos implica asignarles la destreza no solo de moverse de un punto a otro, sino de interpretar su entorno con una sofisticación equiparable a la percepción humana. Esta capacidad de percepción, respaldada por tecnologías avanzadas como sensores, cámaras, radares y sistemas de inteligencia artificial, redefine por completo la dinámica de la conducción al desvincularla progresivamente de la intervención humana directa.

Un área muy importante dentro de este tema es la estimación de ángulos de dirección, un elemento muy importante en el control y la dinámica del vehículo. El tener la representación de una medida angular que describe la dirección en la que se orientan las ruedas delanteras de un automóvil en momento dado, un parámetro importante para determinar la trayectoria y el giro del vehículo. En los modelos de visión más importantes para predicción de ángulos, podemos establecer dos formas que probablemente impliquen aprovechar datos visuales de cámaras, donde podemos designar técnicas de visión clásica, como utilizar filtros, o algún método para detectar el camino[1]. El desafío de mantener un control efectivo se ve por la diversidad de técnicas que se pueden emplear para abordar la estimación de ángulos de dirección. Desde enfoques clásicos basados en controladores PID hasta métodos más avanzados que incorporan redes neuronales y técnicas de aprendizaje profundo [2].

La implementación de modelos complejos y profundos suele demostrar una mayor eficiencia, lo que conduce a realizar un estudio más detallado para la predicción de ángulos de dirección. En el ámbito de las propuestas de aprendizaje profundo, se han explorado diversas arquitecturas, priorizando la precisión y el rendimiento de la red, ya sea en entornos simulados o reales. Dentro de las técnicas más destacadas para la predicción de ángulos de dirección en vehículos autónomos se encuentran las Redes Neuronales Convolucionales (RNC), reconocidas por su calidad superior en el ámbito de la visión artificial. Sin embargo, su implementación se ve desafiada por costos computacionales significativos debido a la profundidad, cantidad de parámetros y capas que componen su arquitectura, especialmente en dispositivos de control con recursos limitados de hardware. Tomando en cuenta esto, el optimizar estos puntos fundamentales en una red, generaría una nueva alternativa en arquitectura.

La neuroevolución es un enfoque revolucionario que combina redes neuronales con algoritmos evolutivos para lograr procesos de aprendizaje y optimización automáticos de arquitecturas neuronales [3], [4]. La neuroevolución utiliza algoritmos genéticos para evolucionar y mejorar las estructuras de las redes neuronales, superando así las limitaciones de métodos convencionales de ajuste de parámetros. Cada individuo en una población evolutiva se representa mediante un conjunto de genes que codifican la arquitectura y los pesos de la red neuronal. La evaluación del rendimiento se basa en su desempeño en tareas específicas, y aquellos mejor adaptados tienen más probabilidades de sobrevivir y reproducirse. La neuroevolución ha encontrado aplicaciones en diversas áreas, desde el control de robots hasta la optimización de arquitecturas de redes neuronales para tareas especializadas.
Este enfoque ofrece una perspectiva única para abordar problemas complejos y dinámicos, permitiendo una adaptación automática a entornos cambiantes. A pesar de los desafíos, como la gestión eficiente de la complejidad de la búsqueda, la neuroevolución destaca como una herramienta prometedora en la creación de sistemas inteligentes y adaptables.

1.1. Definición del problema

En la constante evolución hacia la conducción autónoma, las RNC se han consolidado como herramientas de vanguardia para la predicción de ángulos de dirección en vehículos autónomos. Reconocidas por su excelencia en la interpretación de información visual, las RNC han demostrado una calidad superior en el ámbito de la visión artificial.
Sin embargo, las RNC se ve enfrentada a desafíos prácticos, especialmente en lo que al diseño de su arquitectura y su implementación en dispositivos de control integrados en vehículos autónomos. La complejidad a la arquitectura de las RNC, caracterizada por su profundidad, la cantidad significativa de parámetros y las múltiples capas, resulta en costos computacionales considerables. Este problema se agudiza al considerar dispositivos con recursos de hardware limitados, donde la capacidad de procesamiento se convierte en un factor determinante.
Cuando las RNC cuentan con un número excesivo de parámetros y capas, pueden volverse propensas al sobreajuste, es decir, adaptarse de manera demasiado específica a los datos de entrenamiento sin lograr generalizar eficientemente en situaciones del mundo real. Esta sobredimensión no solo implica mayores requisitos de recursos computacionales, sino que también puede traducirse en una menor capacidad de la red para adaptarse a la diversidad y complejidad de las condiciones de conducción en la práctica.
Al contar con un diseño que evita redundancias innecesarias la RNC no solo mejora su eficiencia en la fase de entrenamiento, sino que también reduce la probabilidad de sobreajuste.
El tener una RNC con menor número de parámetros, proporcionan más robustez y mayor capacidad de generalización ante diversos ambientes cambiantes.

1.2. Propuesta

Implementar una RNC con la capacidad de estimar ángulos de dirección a partir de imágenes obtenidas durante la conducción autónoma. Para lograr este objetivo, se empleará neuroevolución como método de optimización, con el fin de descubrir un modelo que mantenga una precisión adecuada mientras se minimiza el número de parámetros.

1.3. Hipótesis

Mediante la aplicación de la neuroevolución en el diseño de arquitecturas de redes neuronales convolucionales (RNC) para la predicción de ángulos de dirección en situaciones de conducción autónoma, es posible lograr una reducción estadisticamente significativa en la cantidad de parámetros utilizados en la red sin comprometer la precisión en la estimación de los ángulos de dirección. Se espera que la optimización de la arquitectura no solo conduzca a un modelo de RNC con un buen desempeño, sino que también permita una red de baja latencia.

1.4. Justificación

La optimización de parámetros en una Red Neuronal Convolucional RNC se caracteriza por potenciar el rendimiento, la agilidad y la robustez del modelo. Estas mejoras adquieren una relevancia estratégica especialmente en dispositivos de control, donde la eficiencia y la capacidad de respuesta desempeñan un papel fundamental en la toma de decisiones precisa y oportuna. La reducción de parámetros en diseños especializados de arquitecturas dedicada a la estimación de ángulos de dirección es un factor determinante que puede marcar la diferencia en términos de rendimiento y latencia, contribuyendo de manera significativa a la eficacia y eficiencia operativa del sistema en su conjunto.

1.5. Motivación

La motivación de este proyecto es tener un modelo que sea una altenativa para ocupar en un dispositivo de control en vehículos autónomos con menor tiempo de latencia.
En los vehículos autónomos, donde la toma de decisiones precisa y veloz es fundamental, la disminución del tiempo de latencia se convierte en un factor determinante para optimizar la eficiencia operativa y la seguridad del sistema. Este proyecto se motiva al tener una respuesta a la necesidad de contar una ejecución ágil, permitiendo la adaptabilidad a las condiciones cambiantes del entorno de conducción.

1.6. Objetivo general

El objetivo principal de esta investigación es aplicar técnicas de neuroevolución al diseño de arquitecturas de RNC de regresión, para reducir de manera eficiente la cantidad de parámetros necesarios para lograr un rendimiento óptimo en la estimación de ángulos de dirección en imágenes de conducción sin reducir la capacidad de precisión del modelo.

1.6.1. Objetivos específicos

1. Revisar las RNC de predicción de ángulos de dirección más importantes de la literatura.
2. Determinar la base de datos más apropiada para su caso de estudio.
3. Modificar y adecuar un algoritmo de neuroevolución para el diseño RNC de regresión.
4. Calibrar el algoritmo de neuroevolución para redes neuronales de regresión lógica.
5. Determinar una arquitectura RNC para predicción de ángulos de dirección.
6. Comparar la RNC generada con las de la literatura y ver diferencia de parámetros.
7. Generar estadísticas para determinar diferencias significativas.
8. Comparar el tiempo de latencia de las RNCs de la literatura con la RNC generada mediante neuroevolución.

Capítulo 2
Marco referencial

Dando un panorama sobre las investigaciones relacionadas con esta tesis, se abordan enfoques similares que contribuyen al avance en el campo. Los trabajos relacionados pueden dividirse en tres categorías principales:
Primero, investigaciones y propuestas de RNC para la predicción de ángulos de dirección. Diversos estudios han explorado el uso de RNC para la predicción precisa de ángulos de dirección. Estos trabajos se centran en el desarrollo de arquitecturas de redes y algoritmos que mejoran la exactitud y eficiencia en la predicción, proporcionando una base sólida sobre la cual construir aplicaciones avanzadas en este ámbito.
Segundo, optimización de RNC. Otra línea de investigación se enfoca en la optimización de RNC para mejorar su rendimiento, como algoritmos evolutivos. Estos procesos incluyen técnicas como la reducción de la complejidad computacional, la mejora de la eficiencia energética y la aceleración del tiempo de procesamiento. Así como técnicas de aprendizaje por transferencia para el mejoramiento del rendimiento y la calidad de la predicción. El aprendizaje por transferencia ha emergido como una técnica poderosa para mejorar el rendimiento de RNC en diversas aplicaciones, incluida la predicción de ángulos de dirección. Esta técnica permite reutilizar conocimientos de modelos preentrenados en nuevas tareas, acelerando el proceso de entrenamiento y mejorando la precisión de las predicciones.
Finalmente, es esencial abordar la investigación sobre técnicas de compactación y su efecto relevante sobre la latencia en RNC. Este aspecto es fundamental para comprender cómo las técnicas de compactación pueden reducir la latencia sin comprometer la precisión, lo cual es importante para aplicaciones que requieren respuestas rápidas y eficientes. La combinación de estos enfoques ofrece una visión integral y detallada para el desarrollo y mejora de las RNC en la predicción de ángulos de dirección y otras aplicaciones relacionadas.

2.1. Redes Neuronales Convolucionales para estimación de ángulos de dirección

En la literatura actual, se han presentado diversas propuestas de Redes Neuronales Convolucionales (RNC) enfocadas en la conducción autónoma. Entre las contribuciones más destacadas se encuentran las propuestas de Bojarski et al. [5], X. Chen et al. [6], Lee et al. [7], Fang et al. [8], Kim et al. [9], Hwang et al. [10], y Zhang et al. [11]. Cada una de estas propuestas presenta enfoques únicos, empleando diferentes arquitecturas de RNC y conjuntos de datos para abordar los desafíos específicos asociados con la predicción de ángulos de

dirección en entornos de conducción autónoma.
Adicionalmente, estas propuestas emplean conjuntos de datos variados, que pueden ser datos reales de conducción humana y simulaciones. La elección del conjunto de datos tiene un impacto significativo en el rendimiento del modelo, ya que afecta la capacidad de generalización del sistema a situaciones del mundo real.
En algunos casos, los investigadores proporcionan detalles sobre el número total de parámetros en la red, lo que permite evaluar la complejidad y la capacidad de aprendizaje de cada modelo. En la tabla 1 se muestra un pequeño resumen sobre las distintas arquitecturas, donde se utilizan Capa de Normalización (CN), Capa convolucional (Conv) y Capas totalmente conectadas (TC), así como modelos ya existentes pasados a regresión para su expetimentación.

Tabla 1: RNC para estimación de ángulos de dirección

Referencia	Año	Arquitectura RNC	Conjunto de Datos	Número de Parámetros
[5]	2016	CN + 5 Conv + 3 TC	Real	250,000
[6]	2018	5 Conv + 3 TC + Softmax	Real	267,291
[7]	2019	6 Conv + LSTM + 4 TC	Real	–
[8]	2020	RestNet50 + 4 TC	Simulado	25,636,699
[9]	2020	VGG16 + LSTM	Real	–
[10]	2020	4 Conv + 3 TC	Simulación	–
[11]	2021	4 Conv + 4 TC	Real	79,103,537

Dentro de estas arquitecturas podemos destacar a PilotNet [5], una arquitectura profunda que permite mapear directamente las imágenes de una cámara en un vehículo a ángulos de dirección, eliminando la necesidad de diseñar manualmente características. Utiliza datos del mundo real para el entrenamiento, capturando imágenes de carreteras y las acciones de dirección de conductores humanos. La relevancia de PilotNet radica en su capacidad para abordar situaciones diversas, desde cambios de iluminación hasta curvas pronunciadas, allanando el camino para nuevas ideas en la predicción de ángulos y contribuyendo significativamente al avance de la conducción autónoma. La importancia se destaca en su capacidad para manejar desafíos del mundo real, como cambios en la iluminación y curvas abruptas, lo que ha allanado el camino para el desarrollo de modelos más robustos. La idea de utilizar datos del mundo real para el entrenamiento, en lugar de depender exclusivamente de datos sintéticos o simulados, ha influido en la comunidad científica y ha llevado a la búsqueda de enfoques, aplicables en la implementación de sistemas de conducción autónoma. Esta red ha actuado como la base para nuevas ideas y avances en el campo de la conducción autónoma.
También existen diferentes estudios de arquitecturas de predicción de ángulos dirección, como por ejemplo en [11], nos habla sobre la comparación de 3 modelos diferentes de RNC, aplicada en un vehículo eléctrico real, donde en diferentes configuraciones de capas, muestra los resultados de su entrenamiento y validación.

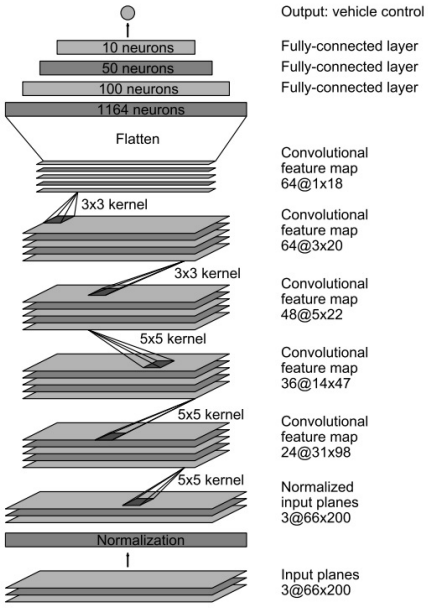

Figura 1: Arquitectura PilotNet [5]

El artículo [12], presenta el desarrollo de un vehículo autónomo a pequeña escala utilizando y estudiando una RNC para la predicción del ángulo de giro. Su objetivo es abordar el error humano, causante del 61 % de los accidentes en Indonesia. Utilizando una Raspberry Pi 4 y una cámara web USB, el prototipo captura imágenes procesadas por la RNC para predecir el ángulo de giro. Se destaca que el modelo Keras de TensorFlow alcanza la autonomía deseada con una precisión RMSE de 0.1145. Aunque el prototipo aún no es apto para carreteras públicas debido a limitaciones de precisión, se sugieren áreas de desarrollo futuro, como la variación de conjuntos de datos y la exploración de métodos alternativos a RNC o optimización para mejorar el rendimiento de predicción del ángulo de dirección.

El artículo [13], aborda el desarrollo de algoritmos para la conducción autónoma, centrándose en el uso de cámaras montadas en vehículos de bajo costo. El enfoque principal es un modelo basado en visión que mapea directamente imágenes de entrada sin procesar a ángulos de dirección utilizando RNC. La aportación es incluir el uso de unidades recurrentes LSTM y Conv-LSTM para combinar señales espaciales y temporales, considerando las observaciones instantáneas de la cámara y los estados históricos del vehículo. Adicionalmente, se utiliza un esquema de retropropagación visual para descubrir y visualizar regiones de la imagen que influyen en la predicción final de la dirección. El estudio experimental se basa en aproximadamente 6 horas de imagenes de conducción humana proporcionados por Udacity [14]. Las evaluaciones cuantitativas demuestran la eficacia y robustez del modelo, incluso bajo condi-

ciones desafiantes como cambios drásticos de iluminación y giros abruptos. En comparación con otros modelos, se destaca su rendimiento superior en la predicción de ángulos de dirección para un vehículo autónomo.

El artículo [15], aborda el avance de la tecnología de conducción autónoma, destacando la transición de sistemas tradicionales complejos a arquitecturas de extremo a extremo basadas en redes neuronales profundas. Estas arquitecturas simplifican la toma de decisiones a bordo de vehículos autónomos, reemplazando múltiples subsistemas con bloques neuronales. El enfoque extremo a extremo, potenciado por técnicas de aprendizaje profundo, utiliza imágenes y datos cinemáticos para predecir acciones de conducción, como velocidad y ángulo de volante. El artículo detalla la implementación de seis arquitecturas para la predicción de velocidad y ángulo de volante, utilizando 78,011 imágenes de escenarios de conducción real. Se destaca la clasificación de arquitecturas según el tipo de datos, incluyendo datos visuales, mezclados y secuenciales. Se menciona la obtención de datos mediante pruebas de conducción con el vehículo autónomo Cloud Incubator Car (CICar), reconociendo la complejidad de las pruebas en condiciones reales y la utilidad de los simuladores de conducción autónoma. El artículo ofrece una visión completa de la evolución tecnológica, resaltando la importancia de grandes conjuntos de datos y la optimización de arquitecturas para la conducción autónoma.

El artículo [16] presenta DeepPicar, una plataforma autónoma de bajo costo basada en redes neuronales profundas que replica en pequeña escala el automóvil autónomo DAVE-2 de NVIDIA. Utiliza una red neuronal propuesta en [5], para predecir ángulos de dirección en tiempo real a partir de imágenes de una cámara frontal. DeepPicar, implementado en una Raspberry Pi 3, demuestra la viabilidad del control en tiempo real de vehículos autónomos asequibles. Se evalúan otras plataformas integradas, incluido el Pi 3, y se destaca la importancia de abordar la contención de recursos compartidos para mantener el rendimiento en tiempo real de la RNC. Este estudio contribuye con una plataforma de prueba accesible para la investigación en vehículos autónomos y ofrece perspectivas valiosas sobre la implementación efectiva de modelos de RNC en entornos de recursos limitados.

2.2. Optimización de Redes Neuronales Convolucionales

Al analizar los diversos estudios y aplicaciones presentes en la literatura relacionada con RNC para la predicción de ángulos, se observa una variedad de enfoques y perspectivas. Algunos investigadores optan por desarrollar nuevas arquitecturas de redes, mientras que otros eligen basarse en aquellas que han demostrado eficacia previamente en esta aplicación específica. Este panorama diverso suscita la pregunta sobre la necesidad de contar con una red que no solo sea robusta, sino también más compacta, lo que implica eficiencia en términos

de recursos computacionales. La inquietud se centra en encontrar un equilibrio óptimo entre la complejidad de la red neuronal y su capacidad para generalizar eficazmente en una amplia gama de situaciones, lo cual es esencial para su aplicabilidad práctica en entornos cambiantes.

En la literatura especializada, se encuentran una variedad de propuestas y enfoques dedicados al mejoramiento y la optimización de redes para predicción de angulos y aplicaciones en vehículos autonomos. Estas propuestas abarcan desde técnicas específicas para algoritmos y modelos hasta metodologías más amplias destinadas a perfeccionar procesos y resultados en distintas disciplinas.

En [17], se centra en el desarrollo de modelos de aprendizaje profundo para la conducción autónoma, específicamente para predecir el ángulo de dirección de un vehículo autónomo. Se comparan tres modelos preentrenados: AlexNet, ResNet18 y DenseNet121, utilizando transfer learning para ajustar los modelos a la tarea de predicción de ángulo de dirección. Los experimentos se realizaron en dos pistas diferentes, y se encontró que ResNet18 y DenseNet121 tenían el menor error porcentual para los valores del ángulo de dirección. En términos de rendimiento, ResNet18 superó a DenseNet121 en precisión, mostrando menos desviación del centro de la pista. Sin embargo, DenseNet121 demostró una mayor adaptabilidad en pistas múltiples, lo que resultó en una mejor consistencia de rendimiento. Además, se menciona que estos modelos fueron evaluados en un sistema embebido de bajo rendimiento Jetson Nano 2GB, y se destaca que ResNet18 y DenseNet121 son adecuados para su implementación en dispositivos embebidos de bajo rendimiento. El estudio aborda la importancia de la conducción autónoma en términos de impacto social y económico, y destaca la reducción de accidentes de tráfico y emisiones de carbono como beneficios potenciales. También proporciona detalles técnicos sobre la arquitectura de los modelos RNC utilizados y describe el proceso de recopilación y aumento de datos.

El articulo [18], nos da una propuesta sobre una RNC creada atravez de inteligencia colectiva, nos describe los algoritmos ocupados para la optimización de una RNC para estimación de angulos en condución autónoma. Nos muestra la diferencia de desempeño que tiene un algoritmo PSO con un algoritmo del murciélago, mostrando la calidad de la predicción através de MSE. Los investigadores utilizaron el conjunto de datos de Udacity [14], para validar el rendimiento de diferentes conjuntos de hiperparámetros y parámetros. La configuración óptima resultó ser: optimizador - Adagrad, tasa de aprendizaje - 0.0052 y función de activación no lineal - unidad lineal exponencial. El estudio destaca que los modelos de aprendizaje profundo muestran resultados superiores pero requieren más épocas de entrenamiento y tiempo en comparación con modelos menos profundos. El enfoque propuesto, que utiliza algoritmos metaheurísticos, demostró mejores resultados en la optimización de RNC en comparación con la sintonización manual.

En [19], proponen un modelo basado en aprendizaje por transferencia. Este modelo combina una RNC utilizando el modelo VGG16. Según los resultados experimentales utilizando un

conjunto de datos de conducción real, se concluye que el modelo propuesto puede predecir eficientemente los ángulos de giro, replicando el comportamiento de conducción humana con un rendimiento superior, mayor precisión y menos tiempo de entrenamiento. Los autores desarrollaron una metodología para predecir el ángulo de dirección simplemente observando las imágenes frontales de un vehículo. Utilizaron un sistema basado en Internet de las cosas (IoT) para recopilar imágenes frontales y ángulos de dirección. Se empleó una cámara Raspberry Pi en conjunto con una unidad de procesamiento Raspberry Pi para capturar imágenes de vehículos, y la unidad de procesamiento se utilizó para recopilar los ángulos asociados con cada imagen.

El artículo [20], discute los resultados de algoritmos de neuroevolución, especialmente Estrategias Evolutivas (ES) y Algoritmo Genético (GA), al abordar el problema de conducción autónoma mediante imágenes de píxeles. Se destaca que, a pesar de la implementación simple de ES y GA, lograron superar a DDQN. Se menciona que la función de optimización de ES necesita mejoras, y existen técnicas para mejorar el rendimiento del GA, como diversas estrategias de cruce y codificación indirecta. El artículo concluye destacando el potencial de los algoritmos de neuroevolución en la tarea de conducción autónoma, sugiriendo que la implementación de técnicas adicionales podría mejorar aún más su rendimiento, incluso en entornos urbanos. La comparación con el algoritmo de aprendizaje por refuerzo DDQN y entre los propios algoritmos evolutivos muestra que estos últimos superan al primero en ciertos aspectos de la tarea, como mantenerse en el centro del carril y recorrer distancias más largas en la pista, lo que sugiere la viabilidad de estos en el campo de los vehículos autónomos.

El artículo [21], se centra en encontrar la arquitectura óptima de las redes neuronales profundas mediante algoritmos genéticos, como se muestra en la Figura 2. La eficacia de las redes profundas en el aprendizaje por imitación depende de la elección adecuada de hiperparámetros, y el diseño manual se considera ineficiente. El estudio utiliza un conjunto de datos de simulación de pista de carreras para entrenar y evaluar modelos. Los resultados revelan mejoras en el rendimiento del aprendizaje por imitación y resaltan la importancia de seleccionar funciones de activación, número de filtros y asignación de capas. La metodología propuesta demuestra ser eficaz para la optimización automática de hiperparámetros en arquitecturas de RNC, destacando su potencial para mejorar la conducción autónoma mediante una neuroevolución eficiente de las redes neuronales profundas. Se concidera importante para futuras investigaciones especificar el número de capas y de filtros en una RNC para su eficiencia.

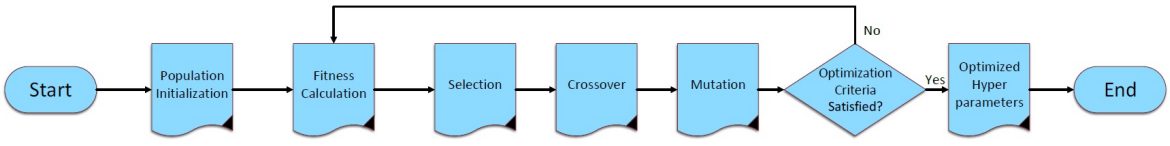

Figura 2: Diagrama de flujo de la técnica de selección basada en GA para problemas de optimización [21].

2.3. Compactación y latencia de Redes Neuronales Convolucionales

Es evidente que la optimización de modelos en RNC es una tarea que va más allá de simplemente mejorar la precisión del modelo. Requiere un análisis exhaustivo que considere no solo la precisión, sino también otras métricas de rendimiento, como la latencia de inferencia y el consumo de recursos. Existen diversos articulos [22]-[25], que buscan comprender factores, estrategias, configuración de la red y la carga del modelo.

En [26], resalta la relación entre el ajuste de hiperparámetros y la optimización de modelos en RNC. Explica que el ajuste de hiperparámetros, que usualmente se enfoca en mejorar la precisión del modelo o minimizar la pérdida, no considera cómo afecta otras propiedades de rendimiento, como la latencia de inferencia y el tamaño del modelo. Esto es importante porque los modelos RNC estándar son grandes y requieren muchos recursos, lo que los hace poco adecuados para dispositivos con recursos limitados, como móviles y dispositivos IoT.

Para abordar este problema, existen técnicas de optimización de modelos, como la poda y la cuantización, que hacen los modelos más pequeños y menos intensivos en computación. Sin embargo, no está claro cómo estas técnicas afectan otras propiedades de rendimiento o cómo interactúan con el ajuste de hiperparámetros. Los resultados muestran que el ajuste de hiperparámetros impacta de manera heterogénea el rendimiento de los modelos RNC y que los modelos optimizados pueden tener un rendimiento significativamente diferente en términos de latencia de inferencia y consumo de batería, incluso si inicialmente tenían un rendimiento similar. Esto sugiere que es importante considerar múltiples propiedades de rendimiento y el efecto de las técnicas de optimización al ajustar y optimizar los modelos RNC para aplicaciones en entornos con recursos limitados.

De igual forma, con la idea de enfocarse en la latencia, en el artículo [27], se centra en la detección, reconocimiento y seguimiento en tiempo real de señales de tráfico para vehículos autónomos, buscando reducir la complejidad computacional y mejorar la eficiencia. Para esto, los autores proponen un enfoque con técnicas de aumento de datos y una RNC de pequeña escala. El artículo se divide en tres fases: detección de señales, utilizando procesamiento de imágenes para identificar regiones de interés; reconocimiento de señales, transformando las regiones de interés en imágenes en escala de grises y procesándolas con una RNC simplificada; y seguimiento de señales, empleando un filtro de Kalman para predecir la posición y tamaño de las señales en fotogramas sucesivos. Los resultados indican que el método es escalable, maneja múltiples clases de señales y reduce la carga computacional, lo cual es esencial para su uso en sistemas embebidos con capacidad de procesamiento limitada.

En técnicas propuestas podemos encontrar [28], donde se proponen dos técnicas para mejorar la eficiencia de RNC en dispositivos IoT con microcontroladores ARM Cortex-M: WPC vec-

toriza los pesos del kernel y DDD reduce la redundancia de datos, disminuyendo la latencia de inferencia.

En este resumen de las diversas investigaciones, se observa que se ha enfocado principalmente en la optimización de hiperparámetros, la comparación de arquitecturas, el aprendizaje por transferencia y el estudio de latencia. Estas líneas de estudio han demostrado resultados positivos en la mejora del rendimiento de los modelos. No obstante, es evidente que ha habido una falta de atención en un enfoque más detallado hacia los parámetros internos de la red y la estructura específica diseñada para abordar la predicción de ángulos de dirección con técnicas que llenen todas estas problemáticas.

La omisión de una atención más profunda a los aspectos internos de la red es un punto de interés en esta investigación. Aunque la optimización de hiperparámetros y la comparación de arquitecturas son importantes, existe una necesidad de explorar y comprender la complejidad de los parámetros internos, así como la idoneidad de la estructura diseñada para la tarea específica en cuestión. Además, se destaca la importancia de la latencia y su investigación. La latencia, es decir, el tiempo de respuesta del modelo, es importante en aplicaciones en tiempo real donde cada milisegundo cuenta. Por lo tanto, dirigir la investigación hacia la búsqueda de una red con un número reducido de parámetros y una baja latencia es fundamental para mejorar la eficiencia y efectividad de los modelos en aplicaciones prácticas.

Aunque la optimización de hiperparámetros y la comparación de arquitecturas son importantes, existe una necesidad de explorar y comprender la complejidad de los parámetros internos, así como la idoneidad de la estructura diseñada para la tarea específica en cuestión. Se destaca la importancia de dirigir la investigación hacia la búsqueda de una red con un número reducido de parámetros.

Capítulo 3
Fundamentos teóricos

3.1. Vehículos autónomos

Un vehículo autónomo, también conocido como vehículo sin conductor o vehículo autodirigido, es un tipo de vehículo que tiene la capacidad de desplazarse y operar sin intervención humana. Estos vehículos utilizan una combinación de tecnologías avanzadas, como sensores, cámaras, radares, láseres y sistemas de procesamiento de datos, para percibir su entorno y tomar decisiones en tiempo real para la navegación y el control. La autonomía de un vehículo autónomo puede clasificarse en diferentes niveles, según la cantidad de control que tiene sobre las funciones de conducción y la necesidad de intervención humana.

3.1.1. Niveles de autonomía

Estos niveles suelen ir desde vehículos con asistencia al conductor hasta vehículos completamente autónomos. El sistema de clasificación más comúnmente utilizado es el definido por la Sociedad de Ingenieros de Automoción (SAE), que va desde el nivel 0 hasta el nivel 5 [29].

Nivel	Nombre	Definición
0	No Conducción Autónoma	Conductor
1	Asistencia al Conductor	Conductor y Sistema
2	Automatización Parcial	Sistema y Conductor
3	Automatización Condicional	Sistema (ADS), Usuario listo
4	Automatización Alta	Sistema
5	Conducción Totalmente Automatizada	Sistema

Tabla 2: Niveles de autonomía de Conducción según la SAE [29].

Como se menciona en la tabla 2, los niveles de autonomía de conducción según la clasificación de SAE proporcionan una estructura que abarca desde la conducción completamente manual en el Nivel 0 hasta la conducción totalmente automatizada en cualquier situación en el Nivel 5. En el Nivel 1, denominado asistencia al conductor, los sistemas realizan tareas específicas, compartiendo responsabilidades con el conductor. Al avanzar al Nivel 2, automatización parcial, el sistema puede controlar tanto la dirección como la aceleración, pero se espera que el conductor complete ciertas sub-tareas y supervise. En el Nivel 3, automatización condicional, el sistema puede conducir en condiciones específicas, pero el usuario debe

estar preparado para intervenir si es necesario. En el Nivel 4, automatización alta, el sistema puede conducir y retroceder en condiciones particulares sin intervención del usuario. Finalmente, en el Nivel 5, se alcanza la conducción totalmente automatizada, donde el sistema puede realizar todas las tareas de conducción en cualquier situación sin requerir la intervención humana. Estos niveles ofrecen una comprensión progresiva de la autonomía vehicular, desde la asistencia parcial del conductor hasta la completa automatización [30].

3.1.2. Sensores y tecnologías de percepción

Los vehículos autónomos dependen de una variedad de sensores para percibir su entorno. Entre los más comunes se encuentran los sensores de radar, cámaras, LiDAR (Light Detection and Ranging), y sensores ultrasónicos. El LiDAR utiliza pulsos de láser para crear un mapa 3D detallado del entorno, lo cual es crucial para la navegación precisa en entornos complejos [31]. Las cámaras en la percepción de los vehículos autónomos están por su capacidad para captar una gran cantidad de información visual del entorno. Se utilizan principalmente para la detección y reconocimiento de objetos, señales de tráfico, peatones y otros vehículos. Las cámaras pueden ser monocromáticas, a color, de alta resolución, o cámaras estéreo que proporcionan información de profundidad.

3.1.3. Procesamiento de datos y algoritmos

El procesamiento de datos en tiempo real es esencial para el funcionamiento de los vehículos autónomos. Los algoritmos aprendizaje automático, como las RNC y los sistemas de aprendizaje profundo, son utilizados para interpretar los datos de los sensores y tomar decisiones de conducción. Estos algoritmos deben ser capaces de manejar grandes volúmenes de datos y realizar predicciones precisas en fracciones de segundo para asegurar una conducción segura y eficiente [32].

3.1.4. Sistemas de control y navegación

Los sistemas de control y navegación son responsables de ejecutar las decisiones tomadas por los algoritmos de procesamiento de datos. Estos sistemas incluyen el control de velocidad, dirección y frenado del vehículo. Además, los sistemas de navegación utilizan datos de GPS, mapas digitales y sensores de entorno para planificar rutas óptimas y evitar obstáculos. La integración de estos sistemas es fundamental para el funcionamiento coordinado y seguro de un vehículo autónomo [33].

3.1.5. Ángulos de dirección

En [34], se menciona que el ángulo de dirección en un vehículo se refiere al ángulo formado por las ruedas delanteras en relación con la línea longitudinal del automóvil cuando se gira el volante. Este ángulo determina la dirección en la que se moverá el vehículo al aplicar la dirección. Al girar el volante, se produce un cambio en el ángulo de dirección, lo que afecta la maniobrabilidad y el control del vehículo. En el proceso de conducción, el ángulo de dirección desencadena la lateralidad del vehículo, influyendo en su movimiento lateral. Lo podemos ver en la figura 3.

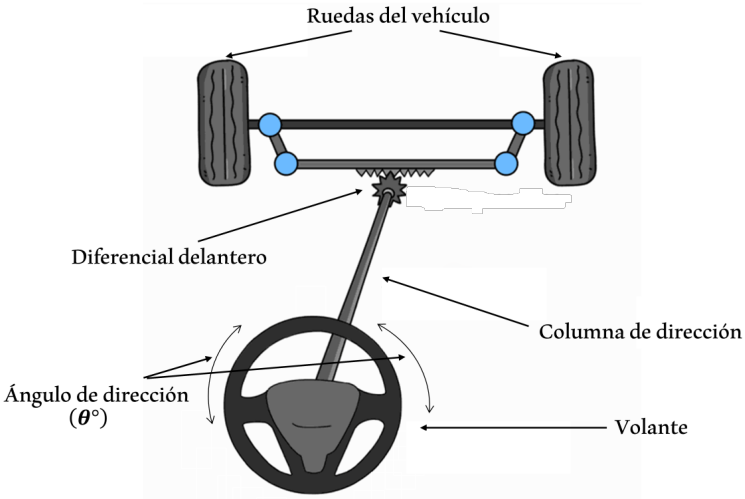

Figura 3: Transmisión angular en un vehículo (imagen tomada y modificada de [35])

La transmisión angular en las ruedas delanteras de un vehículo se realiza a través de la columna de dirección y el diferencial. Cuando se gira el volante como en la figura 4, este movimiento se convierte de rotativo a lineal mediante una caja de dirección que puede contener un mecanismo de cremallera y piñón. Las ruedas delanteras están conectadas al diferencial a través de varillas de acoplamiento y juntas universales. Al girar el volante, estas conexiones transmiten el movimiento a las ruedas delanteras, permitiéndoles girar en la dirección deseada y proporcionando así el control direccional del vehículo [36], [37].

3.1.6. Taxonomía de ángulos de dirección

Para comprender a fondo la diversidad de modelos empleados en la predicción de ángulos en sistemas de conducción autónoma, se exploran los dos enfoques fundamentales que predominan en este ámbito. El primero de estos enfoques implica la meticulosa extracción de coordenadas de límites de carreteras y la aplicación de modelos matemáticos o estadísticos para anticipar el ángulo de dirección. Este método demanda una comprensión profunda de la geometría vial y la capacidad de interpretar las señales visuales, siendo esencial para prever

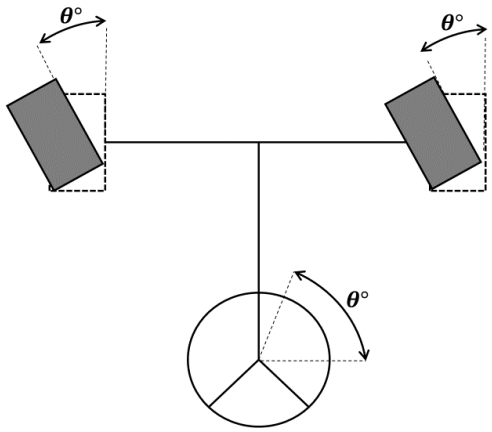

Figura 4: Comportamiento angular de transmisión de un vehículo

el ángulo de dirección en función de la posición relativa de los límites de la carretera [38], [39]. Esto puede ser más susceptible a variaciones en las condiciones ambientales y puede requerir una adaptación constante a cambios en el entorno de conducción.
El segundo enfoque se basa en estrategias de aprendizaje por imitación, haciendo uso de redes neuronales artificiales. Este enfoque innovador implica que el sistema aprenda automáticamente patrones complejos y no lineales directamente de datos de entrada, que suelen consistir en imágenes o secuencias de imágenes. En lugar de depender de una interpretación manual detallada de características específicas de la carretera, las redes neuronales tienen la capacidad única de generalizar y adaptarse a diversas condiciones de conducción. Este enfoque automatizado es valioso para gestionar variaciones en condiciones de iluminación, clima y otros factores ambientales, brindando flexibilidad y robustez al sistema de predicción de ángulos [1], [40].

3.1.7. Desafíos y oportunidades

A pesar de los avances significativos, los vehículos autónomos enfrentan varios desafíos. La seguridad y la fiabilidad son preocupaciones primordiales, ya que los sistemas autónomos deben ser capaces de manejar una amplia gama de situaciones imprevistas. Además, la integración de estos vehículos en la infraestructura de tráfico existente requiere cambios significativos en las políticas y regulaciones. Sin embargo, las oportunidades son igualmente vastas, con el potencial de transformar el transporte, reducir los accidentes de tráfico y mejorar la eficiencia energética [6], [41], [42].

3.2. Aprendizaje automático

El aprendizaje automático o machine learning es una rama de la inteligencia artificial (IA) que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender a realizar tareas sin ser programadas explícitamente para cada una de ellas. En lugar de seguir instrucciones detalladas, los sistemas de aprendizaje automático analizan grandes cantidades de datos, identifican patrones y toman decisiones o predicciones basadas en estos patrones.

3.2.1. Aprendizaje automático supervisado

El aprendizaje supervisado es un enfoque en el campo del aprendizaje automático como se muestra en la figura 5 en donde se entrena un modelo utilizando un conjunto de datos que contiene ejemplos etiquetados. Supervisado significa que durante el proceso de entrenamiento, el modelo recibe información que le indica la relación entre las entradas y las salidas. En el aprendizaje supervisado, el conjunto de datos de entrenamiento consiste en pares de entrada y salida, donde la etiqueta está asociada con cada entrada. El modelo aprende a hacer predicciones o tomar decisiones basándose en la relación entre las entradas y las salidas proporcionadas en el conjunto de datos de entrenamiento.

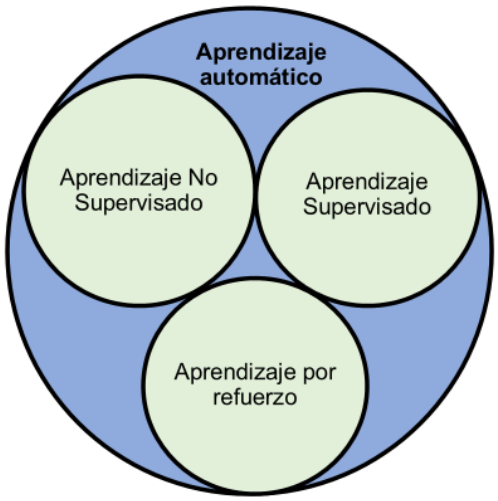

Figura 5: Diagrama de aprendizaje automático

El objetivo final del aprendizaje supervisado es que el modelo pueda hacer predicciones precisas sobre nuevas entradas no vistas después de haber sido entrenado con éxito. Este principio se utiliza en una variedad de aplicaciones, como clasificación de imágenes, reconocimiento de voz, diagnóstico médico, traducción automática y muchas otras tareas de predicción y clasificación.

3.3. Aprendizaje profundo

En este proyecto se ocupa el aprendizaje profundo se centra en el uso de modelos computacionales llamados redes neuronales artificiales para realizar tareas específicas como en la imagen 6 . La característica principal del aprendizaje profundo es la presencia de arquitecturas de redes neuronales profundas, que consisten en múltiples capas de nodos o unidades que realizan transformaciones no lineales de los datos. En lugar de depender de la programación de reglas y características, el aprendizaje profundo permite que el modelo aprenda automáticamente patrones y representaciones complejas a partir de datos de entrada. Las redes neuronales profundas pueden aprender de manera jerárquica, extrayendo gradualmente características más abstractas y complejas a medida que avanzan en las capas. El aprendizaje profundo ha demostrado ser especialmente eficaz en tareas como reconocimiento de imágenes, procesamiento de lenguaje natural, traducción automática, reconocimiento de voz y juegos, entre otros [43]. Algunas de las arquitecturas más conocidas en el aprendizaje profundo incluyen las RNC para el procesamiento de imágenes Y para el procesamiento de secuencias temporales, lo que en vehiculos autónomos es muy aplicativo.

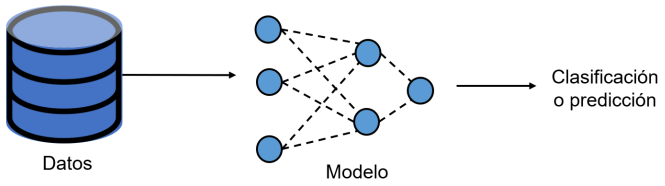

Figura 6: Diagrama de aprendizaje automático

3.3.1. Aprendizaje extremo a extremo

El aprendizaje extremo a extremo es una tarea completa directamente desde la entrada hasta la salida, sin depender de etapas intermedias de procesamiento o extracción de características, a diferencia de otros métodos tradicionales como lo podemos ver en figura 7.

En un sistema de aprendizaje extremo a extremo, el modelo toma datos de entrada crudos y aprende automáticamente las representaciones y características necesarias para realizar la tarea deseada. Este enfoque simplifica el diseño del sistema al eliminar la necesidad de diseñar manualmente pasos intermedios de procesamiento de datos o características, una tarea muy utilizada en vehiculos autónomos como se menciona en [41]. Este aprendizaje ha demostrado ser efectivo en diversas tareas, especialmente en aquellas donde la complejidad del proceso es difícil de modelar explícitamente.

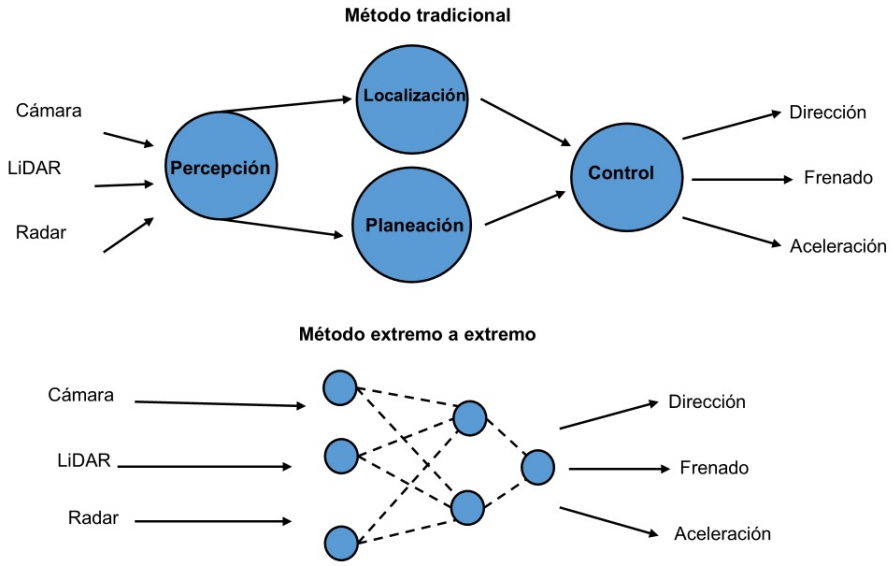

Figura 7: Metodologías de predicciones de ángulos

3.4. Regresión lineal

En la regresión lineal simple, se asume que la relación entre X y Y es lineal, es decir, cualquier cambio en la variable X provoca un cambio proporcional en la variable Y. El modelo estima la mejor línea recta que pueda describir esta relación [44], con el objetivo de predecir los valores de Y basados en los valores de X.

Ecuación:
La fórmula matemática de la regresión lineal simple se expresa como:

$$Y = b_0 + b_1 X + \epsilon \tag{1}$$

Donde:

- Y es la variable dependiente que se desea predecir.
- X es la variable independiente que se usa para hacer la predicción.
- b_0 es el intercepto, que representa el valor de Y cuando $X = 0$. Es el punto donde la línea de regresión corta el eje Y.
- b_1 es la pendiente o coeficiente de regresión, que indica el grado de cambio en Y por cada unidad de cambio en X. Una pendiente positiva implica que Y aumenta a medida que X aumenta, mientras que una pendiente negativa indica que Y disminuye a medida que X aumenta.

- ϵ es el término de error, que captura la variabilidad en Y que no puede ser explicada por X. Representa la diferencia entre los valores observados y los valores predichos por el modelo.

3.4.1. Métricas de evaluación

Las métricas de evaluación en regresión proporcionan medidas cuantitativas del rendimiento de los modelos predictivos en la estimación de valores numéricos [45]. Su importancia radica en su capacidad para ofrecer una evaluación objetiva y comparativa del desempeño del modelo. Estas métricas permiten no solo cuantificar la magnitud de los errores entre las predicciones y los valores reales, sino también identificar patrones específicos de error. Esta información es esencial para ajustar y mejorar el modelo, abordando áreas donde puede haber subestimación o sobre estimación sistemática.
Entre las métricas más utilizadas están:

MAPE

(Mean Absolute Percentage Error, por sus siglas en inglés) MAPE calcula la diferencia porcentual promedio entre las observaciones reales y las predicciones del modelo, tomando en cuenta la magnitud de los errores [46]. La utilización de porcentajes facilita la interpretación de la magnitud de los errores en términos relativos al tamaño de las observaciones reales. Se puede observar en formula 2.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100 \tag{2}$$

donde:

- Y_i es el valor real
- $\hat{Y}_i$ es el valor predecido
- n número de ejemplos

MSE

(Mean Squared Error, por sus siglas en inglés) calcula la magnitud promedio de los errores al cuadrado entre las predicciones del modelo y los valores reales. Elevando al cuadrado los errores, se penalizan de manera más significativa los errores más grandes, lo que refleja una mayor sensibilidad a las discrepancias entre las predicciones y los valores reales.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{3}$$

donde:

- Y_i es el valor real
- $\hat{Y}_i$ es el valor predecido
- n número de ejemplos

MAE

MAE (Mean Absolute Error, por sus siglas en inglés)mide el promedio de los errores absolutos entre las predicciones y los valores reales. Cada diferencia entre la predicción y el valor real se toma en valor absoluto para asegurar que los errores negativos y positivos no se cancelen entre sí.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

donde:

- Y_i es el valor real
- $\hat{Y}_i$ es el valor predecido
- n número de ejemplos

3.5. Redes Neuronales Artificiales

Una Red Neuronal Artificial (RNA) es un modelo computacional inspirado en la estructura y funcionamiento del cerebro humano, diseñado para resolver problemas complejos de clasificación, regresión, y otras tareas relacionadas con el reconocimiento de patrones y el aprendizaje automático. Está formada por una red interconectada de nodos, conocidos como neuronas artificiales o unidades, organizadas en capas.
Estas capas incluyen:
Capa de entrada: Recibe los datos de entrada del modelo. Cada neurona en esta capa representa una característica o variable del conjunto de datos.
Capas ocultas: Procesan la información a través de las conexiones entre las neuronas. En estas capas se llevan a cabo cálculos mediante funciones matemáticas, lo que permite a la red detectar patrones y relaciones complejas entre los datos. Una red puede tener una o más capas ocultas.

3.5.1. Redes Neuronales Artificiales

Las Redes Neuronales Convolucionales (RNC), son un tipo especializado de red neuronal diseñado para procesar y analizar datos bidimensionales, especialmente imágenes. Desde su introducción, las RNC han revolucionado diversas áreas de la visión por computadora y el reconocimiento de patrones debido a su capacidad para aprender características jerárquicas directamente a partir de los datos de entrada [47]. A diferencia de las redes neuronales tradicionales, las RNC aprovechan la estructura espacial de los datos mediante operaciones de convolución y agrupación, lo que permite una extracción de características más eficaz y eficiente [48].

Las RNC son ampliamente utilizadas en aplicaciones que requieren el procesamiento de imágenes, tales como la detección de objetos, clasificación de imágenes, segmentación semántica y conducción autónoma. Esto se debe a su capacidad para manejar grandes cantidades de datos y su eficiencia para identificar patrones complejos en imágenes [49]. Las RNC han sido fundamentales en el desarrollo de sistemas avanzados de inteligencia artificial, como los vehículos autónomos, que dependen en gran medida del procesamiento y análisis de imágenes en tiempo real. En la Figura 8, se muestra el proceso de una RNC. Esta arquitectura es ampliamente utilizada en aplicaciones de visión artificial debido a su capacidad para aprender representaciones jerárquicas de las imágenes.

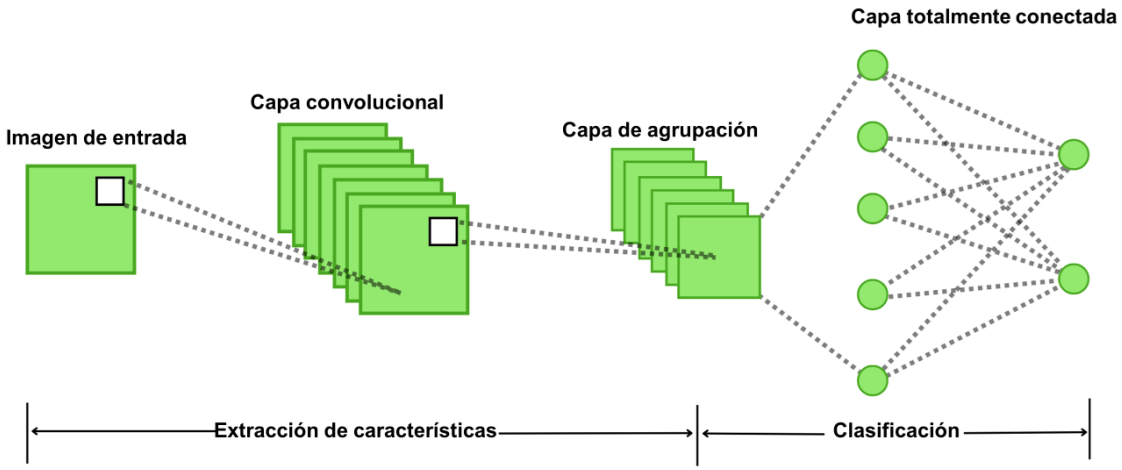

Figura 8: Proceso de una Red Neuronal Convolucional

3.5.2. Capas convolucionales

Un elemento fundamental de las RNC son las capas convolucionales. Estas capas realizan operaciones de convolución, un proceso matemático que permite extraer características significativas de los datos de entrada. La convolución se lleva a cabo utilizando filtros (kernels) que se deslizan sobre la entrada para detectar patrones locales como bordes, texturas y formas. La operación de convolución se puede expresar matemáticamente como:

$$(I * K)(m, n) = \sum_i \sum_j I(m - i, n - j) \cdot K(i, j) \tag{5}$$

- I es una matriz bidimensional que representa la imagen de entrada o cualquier otro conjunto de datos bidimensional.
- K es una matriz bidimensional conocida como el kernel o máscara de convolución. Esta matriz tiene dimensiones más pequeñas que I y se utiliza para aplicar efectos o filtros específicos a la imagen.
- m y n son las coordenadas de la posición en la que se está calculando el valor resultante de la convolución.

Las capas convolucionales son esenciales porque permiten la detección de características jerárquicas. En las primeras capas, los filtros pueden detectar bordes simples y texturas básicas. A medida que se avanza a capas más profundas, los filtros pueden detectar patrones más complejos y específicos, como partes de objetos y estructuras completas. Esto es crucial para la comprensión de imágenes y el reconocimiento de objetos [50]. Adicionalmente, estas capas pueden ser complementadas con técnicas como la normalización por lotes (batch normaliza-

tion) para mejorar la estabilidad y velocidad del entrenamiento de las redes neuronales [51].

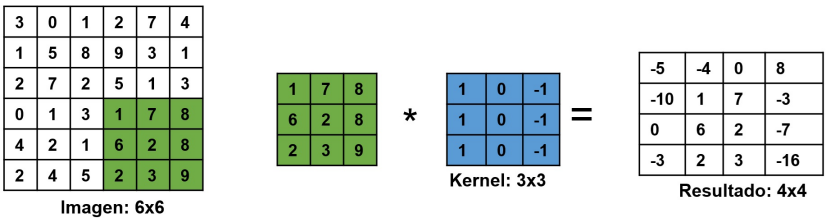

Figura 9: Proceso de convolución

3.5.3. Capas de agrupación

Las capas de agrupación, también conocidas como capas de pooling, siguen a las capas convolucionales y se utilizan para reducir la dimensionalidad de las representaciones de las características, así como para hacer que las redes sean más robustas a pequeñas variaciones en la posición de las características aprendidas. La reducción de dimensionalidad ayuda a disminuir la carga computacional y a prevenir el sobreajuste.

Hay dos tipos comunes de capas de agrupación: la agrupación máxima (max pooling) y la agrupación promedio (average pooling).

- **Agrupación máxima (Max Pooling)**: Selecciona el valor máximo dentro de un parche de la matriz de características. Esto permite conservar las características más prominentes detectadas por las capas convolucionales [52].
- **Agrupación promedio (Average Pooling)**: Calcula el promedio de los valores dentro de un parche de la matriz de características. Aunque menos común que la agrupación máxima, puede ser útil en ciertos contextos para conservar la información de las características.

La operación de pooling se puede expresar matemáticamente como:

$$P(m,n) = \max_{i,j}(I(m+i, n+j)) \tag{6}$$

para max pooling, donde $P(m,n)$ es el valor máximo en el parche de la matriz de características centrado en (m,n). La agrupación promedio se calcula como:

$$P_{avg}(m,n) = \frac{1}{|R|} \sum_{(i,j)\in R} I(m+i, n+j) \tag{7}$$

donde R representa la región de pooling.

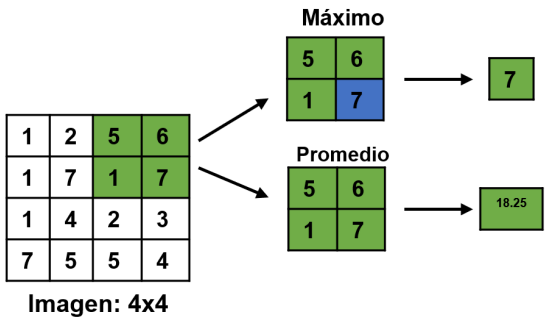

Figura 10: Proceso de Pooling

3.5.4. Capas totalmente conectadas

Las capas totalmente conectadas, conocidas como capas densas, son un componente fundamental de las redes neuronales artificiales. En estas capas, cada neurona o nodo está conectado a cada neurona de la capa anterior y siguiente, creando una red completamente interconectada. Estas capas son responsables de la toma de decisiones finales basadas en las características extraídas y procesadas por las capas anteriores.

La operación en una capa totalmente conectada se puede describir mediante la siguiente fórmula:

$$y_i = f\left(\sum_{j=1}^{n} w_{ij} \cdot x_j + b_i\right) \tag{8}$$

Donde:

- y_i es la salida de la neurona i en la capa.
- f es la función de activación que introduce no linealidades en la red [53].
- w_{ij} son los pesos que conectan la neurona j de la capa anterior con la neurona i de la capa actual.
- x_j es la salida de la neurona j en la capa anterior.
- b_i es el sesgo (bias) asociado con la neurona i.
- n es el número de neuronas en la capa anterior.

Las capas totalmente conectadas suelen aparecer al final de la red y se utilizan para realizar la clasificación final basándose en las características extraídas por las capas convolucionales y de agrupación. Estas capas son esenciales para las decisiones de alto nivel y la interpretación final de los datos procesados [54].

3.5.5. Funciones de activación

Las funciones de activación juegan un papel crucial en las RNC al introducir no linealidades en la red, permitiendo que modelen relaciones complejas en los datos. Algunas de las funciones de activación más comunes incluyen la función sigmoide, la tangente hiperbólica (tanh) y la unidad lineal rectificada (ReLU).

- **Sigmoide**: La función sigmoide transforma la entrada en un valor entre 0 y 1, lo que puede interpretarse como una probabilidad. Sin embargo, su uso en redes profundas es limitado debido al problema de la desaparición del gradiente [55].
- **Tangente hiperbólica (tanh)**: Similar a la sigmoide, pero sus valores de salida varían entre -1 y 1, lo que puede resultar en una convergencia más rápida durante el entrenamiento.
- **ReLU**: La función ReLU es actualmente la más popular debido a su simplicidad y eficacia. Transforma la entrada negativa en cero y deja las entradas positivas sin cambios, lo que ayuda a mitigar el problema de la desaparición del gradiente [56].

3.5.6. Regularización

La regularización es una técnica crucial para prevenir el sobreajuste en las RNC. Existen varias técnicas de regularización que se aplican comúnmente:

- **Dropout**: Consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento para prevenir la co-adaptación excesiva de las unidades [57].
- **Normalización por lotes (Batch Normalization)**: Esta técnica normaliza las salidas de una capa antes de pasarlas a la siguiente, estabilizando y acelerando el proceso de entrenamiento [58].
- **Regularización L2**: Añade una penalización proporcional a la magnitud de los pesos de la red en la función de pérdida, fomentando que los pesos sean pequeños y, por ende, menos propensos al sobreajuste.

3.5.7. Sobreajuste

El sobreajuste (overfitting) es un fenómeno en el aprendizaje automático donde un modelo se adapta excesivamente a los datos de entrenamiento, aprendiendo tanto los patrones generales como el ruido o las particularidades irrelevantes de esos datos. Esto resulta en un modelo que tiene un rendimiento excelente en el conjunto de entrenamiento, pero un desempeño deficiente al enfrentarse a nuevos datos o al generalizar a otros contextos. El sobreajuste ocurre cuando el modelo es demasiado complejo para la cantidad de datos disponibles.

3.5.8. Aplicaciones y avances recientes

Las RNC han demostrado ser una herramienta poderosa en diversas aplicaciones. En el campo de la visión por computadora, las RNC han mejorado significativamente la precisión en tareas de clasificación de imágenes y detección de objetos. Por ejemplo, la arquitectura AlexNet fue pionera en el uso de RNC para ganar el concurso ImageNet en 2012, estableciendo un nuevo estándar en la visión por computadora [49]. En el ámbito de los vehículos autónomos, las RNC se utilizan para el reconocimiento de señales de tráfico, la detección de peatones y otros vehículos, así como para la segmentación de carriles en la carretera. Empresas como Tesla y Waymo han integrado RNC en sus sistemas de conducción autónoma para mejorar la seguridad y la eficiencia de sus vehículos [5], [59].

3.5.9. Latencia en Redes Neuronales

Latencia es el tiempo total que transcurre desde el inicio de una acción hasta la obtención de un resultado. En el ámbito de la informática y las telecomunicaciones, la latencia se refiere al retraso entre la entrada de una solicitud o comando y la respuesta resultante. Es un factor crítico en el rendimiento de redes y sistemas informáticos, afectando la rapidez con la que los datos son procesados y transmitidos. La latencia puede estar influenciada por varios componentes del sistema, incluyendo el hardware, el software y la infraestructura de red.
En [60], la latencia en RNC se refiere al tiempo total necesario para procesar una solicitud de inferencia en dispositivos. La latencia puede afectar significativamente el rendimiento del modelo. Una baja latencia es especialmente importante en aplicaciones que requieren respuestas rápidas, como la conducción autónoma, la realidad aumentada y los dispositivos IoT. La latencia en los modelos profundos es una medida crítica que puede influir en el rendimiento general de un sistema, especialmente en aplicaciones que se requieren. La latencia total se puede desglosar en varios componentes clave: procesamiento en el dispositivo, transmisión de datos y procesamiento en la nube [61].

3.5.10. Factores de Latencia

La arquitectura de una RNC es uno de los principales factores que influyen en la latencia. La profundidad de la red, es decir, el número de capas, y la complejidad, definida por el número de parámetros y operaciones por capa, afectan significativamente el tiempo de procesamiento.

- Profundidad de la Red: Las RNC más profundas, como ResNet-50 o VGG-16, tienen muchas capas, lo que aumenta el tiempo de inferencia debido al procesamiento secuencial de cada capa. Aunque las redes profundas pueden capturar características más complejas y mejorar la precisión, esto da una mayor latencia.

- Complejidad de la Red: Redes con un gran número de parámetros, como Inception-v3, requieren más operaciones de cálculo (multiplicaciones y sumas), lo que incrementa el tiempo de inferencia. La cantidad de parámetros y las operaciones necesarias para calcular las salidas de cada capa afectan directamente la latencia total.

- CPU: Las Unidades Centrales de Procesamiento (CPU) son generalmente más lentas para tareas de inferencia en RNC debido a su arquitectura secuencial y menor capacidad de paralelización. Son adecuadas para aplicaciones donde la latencia no es crítica.

- GPU: Las Unidades de Procesamiento Gráfico (GPU) son más eficientes para la inferencia de RNC debido a su capacidad de procesar múltiples operaciones en paralelo. Esto las hace ideales para aplicaciones en tiempo real.

- TPU: Las Unidades de Procesamiento Tensorial (TPU) están diseñadas específicamente para acelerar cargas de trabajo de aprendizaje automático, incluyendo RNCs. Ofrecen una latencia muy baja y son altamente eficientes para ejecutar grandes modelos de RNC.

Existen diferentes técnicas de optimización que pueden reducir la latencia [62], como la cuantización, reduce la precisión de los números utilizados en los cálculos, generalmente de 32 bits a 8 bits. Al disminuir la cantidad de memoria necesaria, se aumenta la velocidad de procesamiento, lo que reduce la latencia. La poda de redes elimina los parámetros redundantes de la red, reduciendo el tamaño del modelo y, por lo tanto, el tiempo de inferencia. Aunque esta técnica puede disminuir ligeramente la precisión, es efectiva para mejorar la latencia..La capacidad de un sistema para manejar un incremento en la carga de trabajo también afecta la latencia.

3.5.11. Sistemas embebidos

Un dispositivo embebido es un sistema informático especializado diseñado para realizar funciones específicas dentro de un dispositivo más grande. Estos dispositivos combinan hardware y software y están integrados en productos de consumo, industriales, médicos, automotrices, y otros.
A menudo, los dispositivos embebidos son altamente optimizados para cumplir con requisitos de tiempo real, recursos limitados y eficiencia energética, y su funcionamiento suele ser transparente para el usuario. Ejemplos de dispositivos embebidos incluyen electrodomésticos, controladores de motores, cámaras digitales, sistemas de navegación GPS y dispositivos de monitoreo médico.

3.6. Búsqueda de arquitecturas neuronales

La Búsqueda de Arquitecturas Neuronales (NAS, por sus siglas en inglés) es un subcampo del aprendizaje automático que se centra en automatizar el proceso de diseño de arquitecturas de redes neuronales. Este enfoque aborda uno de los desafíos más significativos en el desarrollo de modelos de aprendizaje profundo: la configuración de la estructura óptima de la red neuronal [63]. Tradicionalmente, este diseño ha sido una tarea intensiva en tiempo y recursos, ya que requiere un profundo conocimiento y experiencia en el campo. NAS se propone revolucionar este proceso mediante la automatización, utilizando técnicas avanzadas para explorar el espacio de posibles arquitecturas de manera eficiente y efectiva.

3.6.1. Métodos

Existen diversos métodos que permiten la optimización automática de arquitecturas neuronales, explorando eficientemente grandes espacios de búsqueda para encontrar las configuraciones más efectivas para diversas tareas. Estos métodos incluyen enfoques basados en búsqueda aleatoria, aprendizaje por refuerzo, algoritmos evolutivos, y optimización diferenciable, por mencionar algunos. Cada uno de estos enfoques tiene sus propias fortalezas y limitaciones, lo que hace que la elección del método adecuado dependa de los objetivos específicos y las restricciones de recursos del proyecto. Acontinuación se muestran los métodos más destacados en NAS.

3.6.1.1. Aprendizaje por refuerzo El método de aprendizaje por refuerzo (RL) en la Búsqueda de Arquitecturas Neuronales (NAS) utiliza un agente que explora el espacio de búsqueda de arquitecturas mediante un proceso iterativo de toma de decisiones [64]. Matemáticamente, el agente sigue una política $\pi(a|s)$ que define la probabilidad de seleccionar una acción a dado un estado s.

- **Estado y Acción**: En cada paso t, el agente está en un estado s_t que representa la configuración actual de la arquitectura. Elige una acción a_t que define cambios en la arquitectura, como el tipo y número de capas.
- **Recompensa y Actualización**: La arquitectura generada se entrena y evalúa, produciendo una recompensa R_t basada en su rendimiento. Esta recompensa se utiliza para actualizar la política del agente, ajustando π para mejorar la generación de arquitecturas en futuras iteraciones.
- **Optimización**: El objetivo es maximizar la suma esperada de recompensas acumulativas, ajustando la política para seleccionar arquitecturas que maximicen el rendimiento.

Este enfoque permite al agente aprender y refinar su política de diseño de arquitecturas, explorando de manera eficiente el espacio de búsqueda para encontrar configuraciones óptimas.

3.6.1.2. Algoritmos evolutivos Los algoritmos evolutivos en la búsqueda de arquitecturas neuronales (NAS) optimizan la estructura de redes neuronales mediante un proceso similar a la evolución biológica [65]. Se inicia con una población de arquitecturas aleatorias, cada una evaluada según su desempeño. Las mejores arquitecturas se seleccionan y combinan para crear nuevas arquitecturas, mientras que algunas sufren modificaciones aleatorias. Este ciclo de evaluación, selección, cruzamiento y mutación se repite a lo largo de varias generaciones, refinando continuamente la población hasta encontrar una arquitectura óptima para la tarea específica.

3.6.1.3. Decenso del gradiente El descenso del gradiente en la búsqueda de arquitecturas neuronales (NAS) optimiza automáticamente la estructura de redes neuronales utilizando parámetros diferenciables [66]. Inicializa tanto los pesos de la red como los parámetros de la arquitectura, propaga hacia adelante a través del supernet para calcular la función de pérdida y utiliza el descenso del gradiente para actualizar los pesos y los parámetros de la arquitectura. Finalmente, extrae la arquitectura óptima del supernet basada en los parámetros aprendidos, permitiendo encontrar arquitecturas de redes neuronales óptimas de manera eficiente y escalable.

3.6.2. Algoritmos evolutivos en NAS

La Búsqueda de Arquitecturas Neuronales Evolutiva (ENAS) utiliza algoritmos evolutivos (EAs) para automatizar la búsqueda de arquitecturas de modelos [67]. Los EAs son métodos basados en poblaciones inspirados en la evolución biológica. Estos algoritmos siguen un marco general que incluye las etapas de inicialización, evaluación, reproducción y selección.

En la imagen 11 se ilustran diferentes algoritmos evolutivos según paradigmas evolutivos [65]. Los componentes incluyen el Algoritmo Genético (GA), que aplica cruzamiento y mutación para generar nuevas arquitecturas y selecciona los mejores modelos para la siguiente generación; la Programación Genética (GP), que evoluciona programas completos (redes neuronales) mediante la modificación y mejora de estructuras complejas; la Estrategia de Evolución (ES), que adapta parámetros y estrategias de búsqueda para optimizar iterativamente la arquitectura; y la Evolución Diferencial (DE), que utiliza diferencias entre individuos para guiar la mutación y generación de nuevas arquitecturas.

El proceso evolutivo se lleva a cabo en dos fases: el espacio inicial y el espacio de búsqueda. Primero, se crea una población inicial en un espacio previamente definido. Cada miembro de esta población representa una solución para ENAS, es decir, una arquitectura de red neuronal

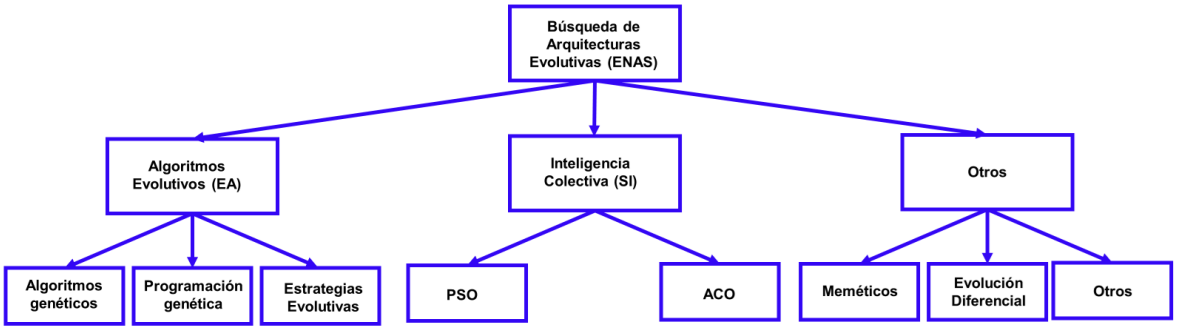

Figura 11: Categoria de algoritmos evolutivos para la optimización según paradigmas evolutivos

profunda. Antes de que las arquitecturas puedan ser parte de la población, deben ser codifica-

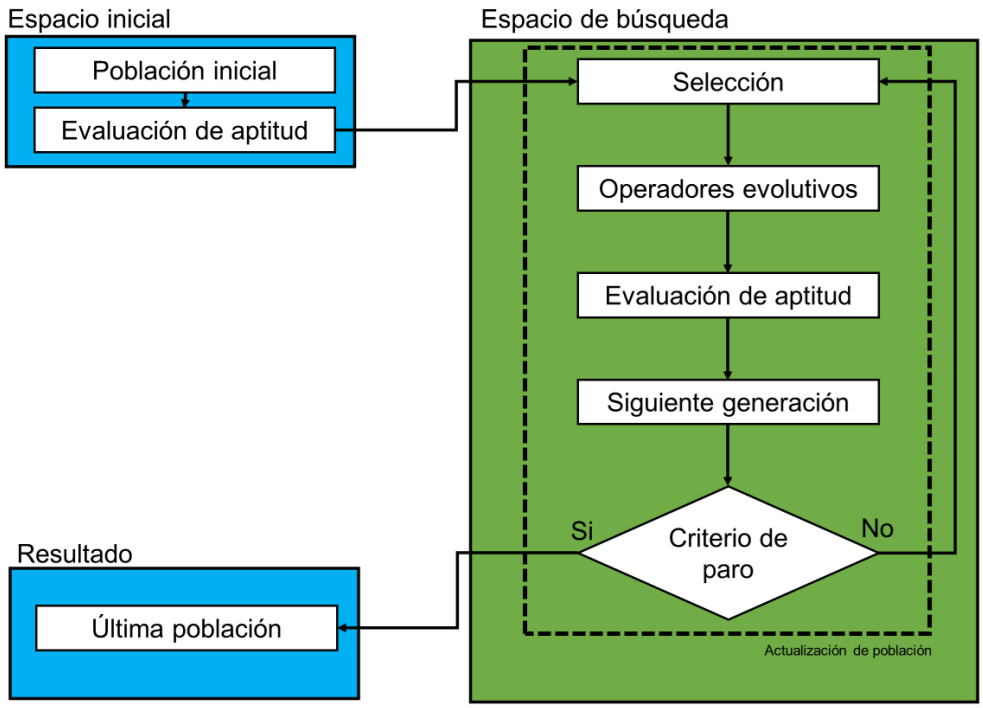

Figura 12: Proceso evolutivo de búsqueda de arquitecturas neuronales

das como individuos. A continuación, se evalúa la aptitud de los individuos generados. Este paso se realiza en dos etapas, utilizando el mismo criterio de evaluación, como precisión, tamaño del modelo o latencia. Una vez evaluada la aptitud de la población inicial, se inicia el proceso evolutivo en el espacio de búsqueda. Durante este proceso, la población se mejora iterativamente mediante selección y operadores evolutivos, como cruza y mutación. Este ciclo de mejora continúa hasta que se cumple un criterio de detención, como alcanzar un número máximo de generaciones o no observar mejoras significativas. Es importante destacar que en algunos paradigmas de computación evolutiva, como la optimización por enjambre

de partículas (SI), la etapa de selección puede no ser necesaria. Finalmente, se obtiene una población que ha completado el proceso evolutivo, representando las mejores arquitecturas profundas encontradas.

3.6.2.1. Sintonización de parámetros En [68], la sintonización en los algoritmos evolutivos es esencial para ajustar los parámetros del algoritmo al problema que se está resolviendo. La sintonización de parámetros implica ajustar los valores antes de ejecutar el algoritmo, mientras que el control de parámetros implica cambiar estos valores durante la ejecución. Los métodos de control de parámetros incluyen:

1. Control determinístico: Los parámetros cambian según una regla predefinida a lo largo del tiempo.

2. Control adaptativo: Los parámetros se ajustan en función del desempeño del algoritmo durante la ejecución.

3. Control auto-adaptativo: Los parámetros se codifican como parte de los individuos en la población y evolucionan junto con ellos.

3.6.3. Neuroevolución

La neuroevolución es un paradigma de aprendizaje que utiliza algoritmos evolutivos para optimizar directamente la arquitectura y los parámetros de redes neuronales [69]. A diferencia de los métodos tradicionales de ajuste de pesos mediante descenso de gradiente, la neuroevolución comienza con una población inicial de arquitecturas neuronales que evolucionan a lo largo de generaciones. Cada arquitectura se evalúa en función de su rendimiento en una tarea específica, y las más exitosas se seleccionan para reproducirse y generar una nueva generación mediante operaciones genéticas.
La neuroevolución encuentra su inspiración en el principio de evolución biológica, el cual implica la transformación gradual de las características de individuos dentro de una población a lo largo de generaciones, con el propósito de adaptarse eficientemente a su entorno.
El proceso de neuroevolución se puede resumir en la creación de una población inicial de redes neuronales individuales mediante una codificación neuronal específica. A partir de esta población, se generan redes reales, las cuales son evaluadas mediante una función de aptitud que mide la calidad de los resultados obtenidos. Luego, se seleccionan las redes más efectivas, se introducen cambios aleatorios para generar una descendencia, y finalmente, se elige la nueva población que continuará con el ciclo de evolución, como se muestra en la Figura 13. [70].

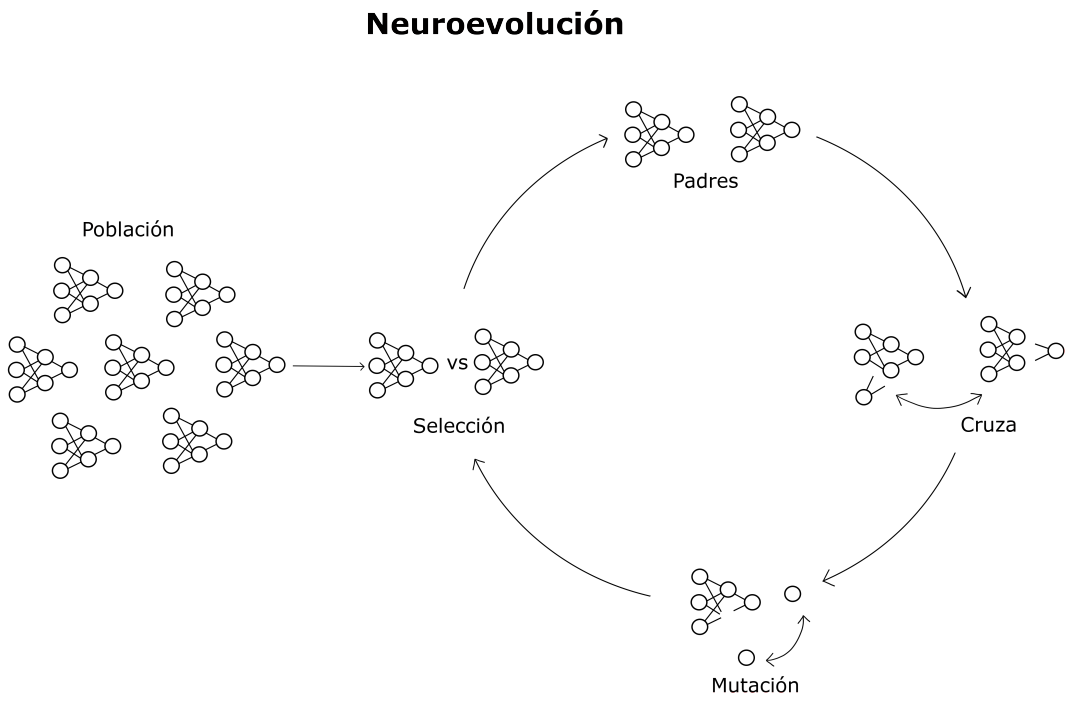

Figura 13: Proceso de neuroevolución

3.6.4. DeepGA

Deep Genetic Algorithm (DeepGA) es un algoritmo evolutivo utilizado para optimizar Redes Neuronales Convolucionales en aplicaciones de reconocimiento visual, como la clasificación de condiciones pulmonares en imágenes de rayos X [71]. Este algoritmo propone una técnica de codificación híbrida compacta para representar las complejas estructuras de estas redes neuronales profundas.

La idea principal detrás de DeepGA es crear automáticamente la arquitectura de la red neuronal mediante la aplicación de algoritmos genéticos, que se inspiran en los procesos naturales de evolución. En lugar de diseñar manualmente la estructura de la red, DeepGA utiliza un enfoque de búsqueda automática basado en la evolución. La investigación compara la eficacia de la codificación híbrida propuesta en DeepGA con otra basada en DenseBlocks, un tipo específico de bloque de construcción en arquitecturas de RNC. La evaluación se centra en cómo estas representaciones afectan la búsqueda de arquitecturas, especialmente en términos de tamaño y rendimiento de la red. El objetivo final de DeepGA en este estudio es encontrar arquitecturas de RNC que sean altamente efectivas en la tarea específica de clasificar condiciones pulmonares en imágenes de rayos X. La codificación de DeepGA se compone de dos

niveles basados en la codificación de Wang [72]. En el primer nivel, denominado Bloques, se definen dos tipos principales:

1. **Bloques Convolucionales:** Estos bloques encapsulan las características de una capa convolucional mediante una única capa convolucional y una operación de pooling opcional. La capa convolucional especifica el número y tamaño de los filtros, utilizando un stride de 1 y zero-padding de 1. Se permite el uso de max pooling, average pooling, o ninguna operación de pooling. La función ReLU y normalización se aplican inmediatamente después de la convolución.

2. **Bloques Fully-Connected:** Estas capas totalmente conectadas se colocan al final de la sección de extracción de características (Bloques Convolucionales) y se describen por el número de neuronas. Utilizan la función ReLU como función de activación, y siempre se añade un último bloque fully-connected con un número de neuronas igual a la cantidad de clases, utilizando la función de activación softmax.

En el segundo nivel, llamado conexiones, se emplea una cadena binaria que define la conectividad entre bloques convolucionales. Cada bit representa la conectividad de una capa anterior no consecutiva, empezando desde el tercer bloque. Este enfoque permite una representación eficiente y compacta de la arquitectura de la red, facilitando así la evolución y optimización automática de las RNCs para tareas específicas, como la clasificación de condiciones pulmonares en imágenes de rayos X.

3.7. Estadísticas de comparación

3.7.1. Kolmogorov-Smirnov

La prueba de Kolmogorov-Smirnov es una prueba no paramétrica utilizada para comparar la distribución de dos muestras o para evaluar si una muestra sigue una distribución específica. Se basa en la distancia máxima entre las funciones de distribución empíricas (CDF) de las muestras comparadas. Calcula el valor D, que es la máxima diferencia entre las dos distribuciones.

Para dos muestras X y Y:

$$D = \max |F_X(x) - F_Y(x)| \tag{9}$$

donde F_X y F_Y son las funciones de distribución acumulativa de las muestras X y Y.

- Hipótesis nula (H0): Ambas muestras provienen de la misma distribución.
- Hipótesis alternativa (H1): Las muestras provienen de distribuciones diferentes.

Algorithm 1 DeepGA

Require: Una población P de N individuos. El número de generaciones T, tasa de cruce $CXPB$, tasa de mutación $MUTPB$, tamaño de torneo $TSIZE$.

Ensure: Inicializar la población (entrenar las redes).

$t \leftarrow 1$
while $t \leq T$ **do**
    Seleccionar $N/2$ padres mediante selección de torneo probabilística.
    $Offs \leftarrow \{\}$
    while $|Offs| < N/2$ **do**
        Seleccionar dos padres aleatorios p_1 y p_2.
        if random(0,1) $\leq CXPB$ **then**
            $O1, O2 \leftarrow$ Crossover(p_1, p_2) ▷ Cruce
        end if
        if random(0,1) $\leq MUTPB$ **then**
            Mutation($O1$, $O2$) ▷ Mutación
            fitness($O1$, $O2$) ▷ Evaluación
        end if
        $Offs \leftarrow Offs \cup \{O1, O2\}$
    end while
    $P \leftarrow P \cup Offs$
    Seleccionar los mejores N individuos en P como sobrevivientes.
    $t \leftarrow t + 1$
end while

3.7.2. Prueba de Kruskal-Wallis (Kruskal-Wallis Test)

La prueba de Kruskal-Wallis es una prueba no paramétrica que compara las medianas de tres o más grupos independientes. Asigna rangos a todas las observaciones de todos los grupos. Calcula un estadístico H que se basa en las sumas de los rangos de cada grupo.

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \tag{10}$$

donde:

- N es el número total de observaciones,
- k es el número de grupos,
- R_i es la suma de los rangos del grupo i,
- n_i es el número de observaciones en el grupo i.
- Hipótesis nula (H0): Todas las muestras tienen la misma mediana.
- Hipótesis alternativa (H1): Al menos una de las muestras tiene una mediana diferente.

3.7.3. Prueba de Mann-Whitney U (Mann-Whitney U Test)

La prueba de Mann-Whitney U es una prueba no paramétrica que evalúa si hay diferencias significativas entre dos grupos independientes, comparando sus distribuciones. Se asignan rangos a las observaciones de ambos grupos y se calcula el estadístico U basado en la suma de rangos.

$$U = R - \frac{n(n+1)}{2} \tag{11}$$

donde:

- R es la suma de los rangos del grupo más pequeño,
- n es el número de observaciones en el grupo.

Otra forma de calcular U es:

$$U = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{12}$$

donde:

- n_1 y n_2 son los tamaños de los dos grupos,
- R_1 es la suma de rangos del grupo 1.
- Hipótesis nula (H0): Las dos muestras provienen de la misma distribución.
- Hipótesis alternativa (H1): Las dos muestras provienen de distribuciones diferentes.

Capítulo 4
Metodologías

4.1. Comparación RNC

La metodología propuesta en Figura 14, se centra en el desarrollo de una RNC que sea capaz de destacarse en un número inferior de parámetros en comparación con las arquitecturas ya existentes, consiste en lograr una calidad de predicción eficiente en la estimación de ángulos de dirección, al mismo tiempo que se busca reducir el número de parámetros en la arquitectura de la red. Esta estrategia responde a la necesidad de optimizar las redes neuronales convolucionales, que a menudo se enfrentan al desafío de mantener un equilibrio entre la complejidad del modelo.

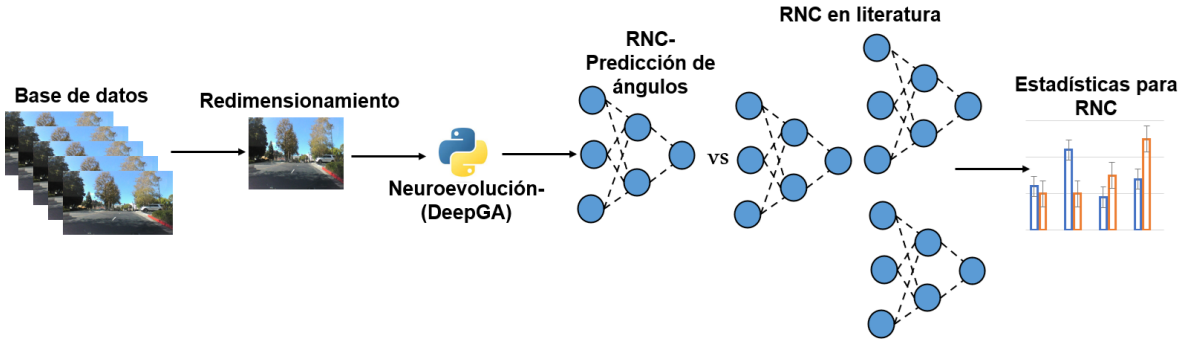

Figura 14: Esta metodología utiliza un conjunto de datos redimensionado a 256x256, que se introduce en un algoritmo de neuroevolución para obtener una arquitectura de RNC. Luego, esta arquitectura se compara con las principales RNC en la estimación de ángulos de dirección. Finalmente, se realizan comparaciones estadísticas para detectar diferencias significativas.

4.1.1. Conjunto de datos

El conjunto de datos utilizado en esta investigación es la *PilotNet*, del repositorio *PilotNet: End to End Learning for Self-Driving Cars* [73], que consta de 45,567 imágenes secuenciales. Estas imágenes fueron capturadas a una tasa de 0.176 fotogramas por segundo durante un total de 72 horas de conducción. Esta tasa de captura implica que, en promedio, se obtuvieron aproximadamente 5.68 imágenes por segundo, asegurando un registro continuo y detallado del entorno visual. Este proceso de captura se realizó en diversas condiciones de conducción, incluyendo variaciones en iluminación, clima y tráfico, para asegurar la diversidad y representatividad del conjunto de datos.

En [5], los autores detallan un procedimiento que consiste en la instalación estratégica de una cámara en un punto clave del vehículo, cuidadosamente seleccionado para maximizar

la calidad y la relevancia de los datos recopilados. Esta cámara está configurada para capturar secuencias de imágenes a una tasa constante y definida de fotogramas por segundo, lo que permite un registro detallado y continuo del entorno visual durante todo el trayecto del vehículo. Este enfoque garantiza la captura exhaustiva de todos los elementos visuales clave que afectan la dinámica de conducción, resultando en un conjunto de datos robusto y valioso para el análisis, como se ilustra en la Figura 15.

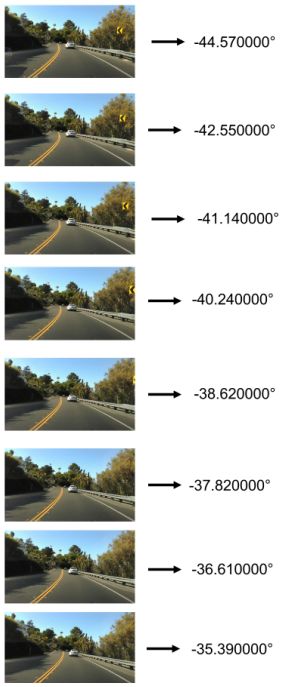

Figura 15: Ejemplos de imágenes de la base de datos con sus respectivos ángulos de dirección del dataset *PilotNet*, del repositorio *PilotNet: End to End Learning for Self-Driving Cars* [73]

De manera complementaria, se incorpora un giroscopio en el volante del vehículo. Este giroscopio registra los cambios en la orientación del volante con una frecuencia equivalente a la captura de imágenes. De este modo, se obtiene un conjunto de datos relacionados, donde cada imagen capturada se asocia con la información del ángulo de dirección registrado por el giroscopio en el mismo instante temporal como en la Figura 16

En la literatura especializada, existen diversos conjunto de datos, donde la más destacada es Udacity [14]. También existe una versión en el repositorio de la IEEE [74].
De igual forma, la conversión de grados a radianes en experimentos relacionados con el ángulo de dirección se lleva a cabo no solo por conveniencia matemática, sino también como una práctica estandarizada en la literatura especializada. Esta estandarización asegura una comprensión y comparación más coherente del comportamiento entre diferentes estudios. Al utilizar radianes, se logra una mayor consistencia con las convenciones matemáticas y trigonométricas comúnmente aceptadas, simplificando así los cálculos y garantizando la compati-

Figura 16: Imágenes con ángulos de dirección asignados, calculados en función de las características visuales del giro y la trayectoria de conducción.

bilidad con bibliotecas y herramientas de programación que utilizan esta unidad angular.

4.1.2. Neuroevolución (DeepGA)

Para poner diseñar nuestra red se utilizó un algoritmo de neuroevolución propuesto en [71], donde su enfoque principal es en RNC de clasificación. En este caso, se hicieron modificaciones en este algoritmo para generar un diseño para redes de regresión, cambiando los criteros de maximización a minimización, al tener el objetivo de encontrar redes que tengan el número más bajo en MAPE (equación 2), donde un MAPE entre 10 % a 20 % se considera un buena predicción [75]. La adaptación del algoritmo se centró en encontrar redes eficientes en términos de su rendimiento predictivo y la cantidad de parámetros utilizados. Buscamos no solo precisión en las predicciones, sino también eficiencia en el uso de recursos. Esto se alinea con la idea de obtener modelos que sean lo más compactos y efectivos posible. Con la idea de generar más diversidad, se optó por elegir tener una variante en el elistismo, tomando al mayor de los menores. Dentro de las adaptaciones realizadas en el marco del algoritmo DeepGA [71], se llevó a cabo una transformación en la entrada de datos. La meta principal de esta modificación fue establecer una asociación directa entre el ángulo de dirección y cada imagen en el conjunto de datos.

4.1.3. RNC-predicción de ángulos

Este proceso implica someter la red a un mayor número de épocas, aumentar el número de épocas permite que la red se ajuste mejor a los datos de entrenamiento y, en consecuencia, proporciona una oportunidad para observar cómo la arquitectura se adapta y mejora con el tiempo. Durante este proceso de reentrenamiento, se da una atención especial a la métrica de la fórmula correspondiente que presenta en la ecuación 8. El MSE mide la diferencia cuadrática promedio entre las predicciones de la red y los valores reales, proporcionando así una medida de la precisión de la red en términos de la magnitud de los errores. La minimización

del MSE implica que la red está mejorando en su capacidad para hacer predicciones más precisas y se está aproximando de manera más efectiva a la tarea que se le ha encomendado. De igual, forma verificar el número de parametros encontrados. Como se menciona en el capítulo 6 de [76], al diseñar arquitecturas de RNC, se debe considerar cuidadosamente tanto la profundidad como el número de parámetros. La elección de modelos más profundos no solo implica agregar complejidad, sino que también refleja la capacidad del modelo para expresar creencias sobre la estructura de la función que está aprendiendo.

4.1.4. Propuestas RNC-estimación de ángulos

Como se muestra en la Tabla 1 del marco referencial 2, existen diversas propuestas de RNC, tomando como base estas redes, se propone llevar a cabo un proceso de entrenamiento de estas arquitecturas. La base de datos será la misma mencionada en el repositorio [73] con 10,000 imágenes. Se establecerán hiperparámetros comúnes, como el número de imágenes, épocas y tamaño de lote (batch), con el objetivo de crear un entorno de comparación equitativo para todas las arquitecturas. Interpretar estas arquitecturas con capas LSTM [7], o tomando algunas preentrenadas pasadas a regresión como VGG16 [9], RestNet50 [8], nos darán resultados ocupando un número de parámetros diferentes en cada red, con resultados similares. Al utilizar la misma base de datos y configuración de entrenamiento, se podrán destacar las fortalezas y debilidades de cada propuesta, proporcionando una visión de su desempeño en la tarea específica de predicción de ángulos.

Tabla 3: RNC de comparación

Referencia	Año	Arquitectura RNC	Conjunto de Datos	Número de Parámetros
[5]	2016	5 Conv + 3 TC	Real	250,000
[8]	2020	RestNet50 + 4 TC	Real	25,636,699
[9]	2020	VGG16	Real	138,357,456
[9]	2020	VGG16 + LSTM	Real	156,223,443
[18]	2022	13 Conv + 4 TC	Real	1,335,758

4.1.5. Estadísticas RNC

Después de haber resaltado las variaciones en los parámetros de las distintas redes neuronales, se procede a realizar un análisis más profundo para determinar si estas diferencias son estadísticamente significativas. Fueron llevadas a cabo 10 ejecuciones completas de todas las redes, generando conjuntos de datos con el propósito de evaluar la normalidad de su distribución y decidir si se deben aplicar pruebas paramétricas o no paramétricas. La prueba de $Kolmogorov - Smirnov$ se emplea para verificar la normalidad de los datos. Posteriormente, se realiza una prueba de $Kruskal - Wallis$ para determinar si existen diferencias significativas entre las medianas de los conjuntos de datos generados por cada red. Este análisis global permite evaluar si alguna de las arquitecturas presenta un rendimiento estadísticamente distinto en comparación con las demás. Para identificar de manera más específica en qué conjuntos de datos se encuentran las diferencias significativas, se emplea la prueba de $Mann-WhitneyU$. Esta prueba se aplica entre pares de conjuntos de datos para determinar si hay diferencias significativas en las medianas. El objetivo es identificar qué arquitecturas presentan divergencias estadísticas en términos de su desempeño. Estas pruebas permiten establecer conclusiones más sólidas sobre la eficacia relativa de cada arquitectura en el contexto del proyecto.

4.2. Latencia

La metodología mostrada en la Figura 17 tiene el proposito de comparar la latencia en RNC para la estimación de ángulos de dirección, en vehículos autónomos se presenta en varias etapas clave.

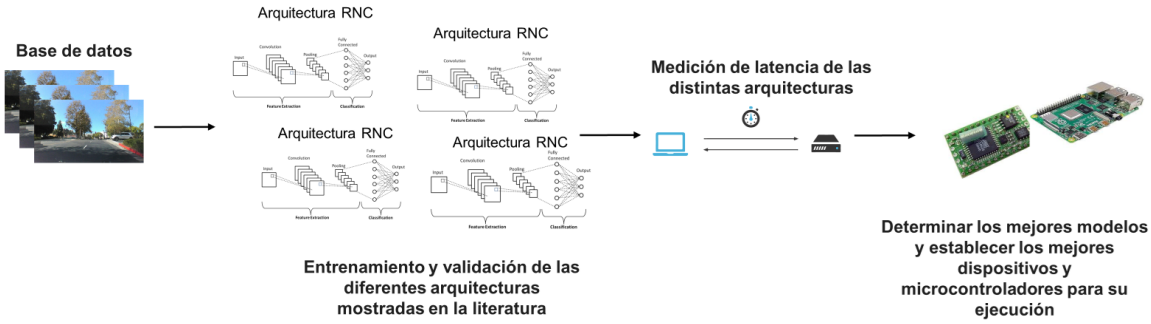

Figura 17: Metodología de comparación de latencia en RNC para estimación de ángulos de dirección

En la metodología presentada en la Figura 17 para la comparación de latencia en Redes Neuronales Convolucionales (RNC) para la estimación de ángulos de dirección, se ocupa la misma base de datos utilizada en la metodología de comparación de la imagen 14.
Esta base de datos está compuesta por imágenes capturadas desde un vehículo, que sirven como datos de entrada para el entrenamiento de varias arquitecturas de RNC reportadas en la literatura. Cada una de estas arquitecturas es entrenada y validada con el objetivo de determinar su precisión en la estimación de los ángulos de dirección a partir de las imágenes. Una vez completado el proceso de entrenamiento y validación, se procede a medir la latencia de cada arquitectura. La latencia se define como el tiempo, medido en milisegundos, que una arquitectura de RNC tarda en procesar una imagen y estimar el ángulo de dirección correspondiente. Tras la medición de la latencia, se comparan los resultados obtenidos para identificar los modelos de RNC que ofrecen un mejor desempeño en términos de precisión y menor latencia. Este análisis comparativo permite determinar cuáles son las arquitecturas más eficientes para la tarea específica de estimación de ángulos de dirección. Finalmente, basándose en los resultados de precisión y latencia, se recomienda el uso de los mejores modelos de RNC junto con los dispositivos y microcontroladores más adecuados para su implementación. La selección de estos dispositivos se realiza considerando aquellos que proporcionen la mayor eficiencia y el mejor rendimiento. Esta metodología no solo permite comparar y seleccionar las mejores arquitecturas de RNC para la estimación de ángulos de dirección, sino también determinar cuál es el microcontrolador más apropiado para su implementación, optimizando así el desempeño del sistema en aplicaciones de vehículos autónomos. Esto asegura que el sistema final no solo sea preciso en la estimación de los ángulos de dirección, sino también eficiente en términos de tiempo de procesamiento y recursos de hardware utilizados.

Capítulo 5
Experimentos y resultados

En esta sección se muestran los experimentos y los resultados que se han obtenido con base a la metodología.

5.1. Banco de imágenes

En el caso del banco de imagenes en este experimento, se realiza un redimencionamiento de 255 x $455 \rightarrow 255$ x 255 esto, con la idea de poder implementarlo al algoritmo de neuroevolución, tratando de reducir los tiempos computacionales. Al redimensionar la resolución de las imágenes de 455 píxeles de alto a 255 píxeles, se puede lograr un ahorro significativo en términos de recursos computacionales, como tiempo y memoria, al ejecutar el algoritmo de neuroevolución. Esto puede ser beneficioso para acelerar la velocidad de entrenamiento y mejorar la eficiencia del proceso sin quitar demasiada información visual relevante para la predicción de ángulos. Utilizando la siguiente frecuencia de datos. Para una mejor comprensión de los resultados fueron expresados en radiantes a través de $\theta_{\text{radianes}} = \left(\frac{\pi}{180}\right) \cdot \theta_{\text{grados}}$

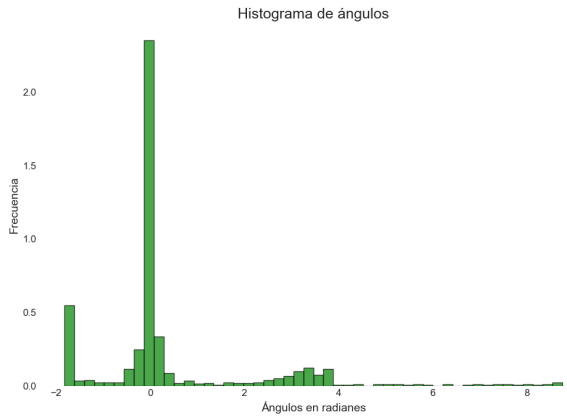

Figura 18: Histograma de freciencia de ángulos en las imágenes

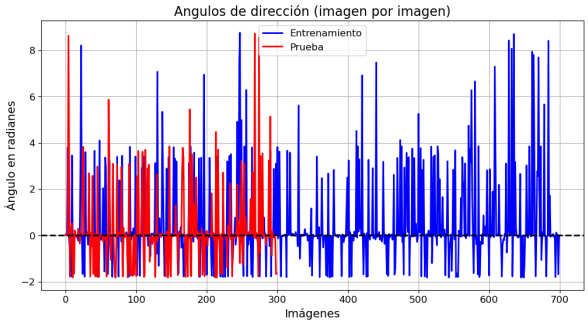

Figura 19: Frecuencia de ángulos a lo largo de las imágenes

5.2. Neuroevolución

En el proceso de construcción de redes neuronales en la población inicial de DeepGA, se han establecido una serie de parámetros y hiperparámetros que guiarán la generación inicial de estas arquitecturas. Estos parámetros se detallan en la tabla 4 donde se especifican, como el número mínimo y máximo de capas convolucionales y capas totalmente conectadas, así como la variedad de opciones para el número de filtros, tamaños de filtros, tipos de pooling, tamaños de pooling y número de neuronas. Los parámetros fueron definidos según los criterios establecidos en [71]. En particular, se fijaron los valores máximos y mínimos para el número de capas convolucionales y totalmente conectadas para restringir el espacio de búsqueda y enfocar la exploración únicamente en arquitecturas con esas especificaciones.

Parámetro	**Valores**
Mínimo de Capas Convolucionales	2
Máximo de Capas Convolucionales	6
Mínimo de Capas Totalmente Conectadas	2
Máximo de Capas Totalmente Conectadas	4
Número de Filtros	2, 4, 8, 16, 32
Tamaño de Filtros	2, 3, 4, 5, 6, 7, 8
Tipo de Pooling	Max, Avg
Tamaño de Pooling	2, 3, 4, 5
Número de Neuronas	4, 8, 16, 32, 64, 128
Épocas de entrenamiento	30
Tasa de aprendizaje (Optimizador Adam)	1×10^{-4}
Tamaño de lote	10

Tabla 4: Hiperparámetros de inicialización en las RNC

Se optó por los parámetros en el algoritmo genético de la Tabla 5, se introducen los parámetros que han sido calibrados mediante un grid search de la tabla 7.

Parámetro	Valores
Tamaño de la Población	20
Número de Generaciones	30
Torneo	5
Porcentaje de Cruza	0.9
Porcentaje de Mutación	0.7

Tabla 5: Parámetros del algoritmo genético

5.3. Sintonización de parámetros

Para encontrar la sintonización correcta en el algoritmo genético, DeepGA, se optó por utilizar un cuadro de búsqueda (Grid Search), que permite encontrar los parámetros del algoritmo mediante la evaluación exhaustiva de todas las combinaciones posibles de valores predefinidos. En esta sintonización, se llevo acabo el entrenamiento de las RNC con los parámetros de la Tabla 6

Parámetro	Valores
Tamaño por lote	10
Épocas	30
Porcentaje de entrenamiento	0.0001
Número de imagenes de entrenamiento	800
Número de imagenes de prueba	200

Tabla 6: Parámetros utiliados en entrenamiento de RNCs

Se consideran varios parámetros clave para equilibrar el rendimiento del modelo:

1. Tamaño de la Población: Se probaron valores como 10, 20 y 30 para encontrar un equilibrio entre diversidad de soluciones y tiempo de cómputo.

2. Porcentaje de Mutación: Se exploraron porcentajes de 0.1, 0.3, 0.5 y 0.7 para ajustar la capacidad de exploración del algoritmo y su estabilidad.

3. Porcentaje de Cruza: Se evaluaron valores de 0.6, 0.7, 0.8 y 0.9 para combinar de manera efectiva las características de las soluciones padres.

4. Número de Generaciones: Se consideraron 10, 20 y 30 generaciones para balancear el tiempo de cómputo con la mejora en la calidad de las soluciones.

Estos experimentos fueron llevados acabo en una laptop DELL de 16 RAM, y una GPU 3050, llevando un tiempo aproximado de 362 horas que corresponden 15 días aproximadamente. En la Tabla 7, se observa una tendencia de resultados óptimos con combinaciones altas de mutación y cruza, mientras que la población y el número de generaciones tienen un impacto menor en la diversidad de exploración. La configuración más eficiente es la número 96.

Tabla 7: Grid Search

Configuración	Población	% Mutación	% Cruza	Generaciones	MAPE
1	10	0.1	0.6	10	0.1500
2	10	0.1	0.6	20	0.1450
3	10	0.1	0.6	30	0.1400
4	10	0.1	0.7	10	0.1550
5	10	0.1	0.7	20	0.1500
6	10	0.1	0.7	30	0.1480
7	10	0.1	0.8	10	0.1600
8	10	0.1	0.8	20	0.1550
9	10	0.1	0.8	30	0.1500
10	10	0.1	0.9	10	0.1650
11	10	0.1	0.9	20	0.1600
12	10	0.1	0.9	30	0.1550
13	10	0.3	0.6	10	0.1700
14	10	0.3	0.6	20	0.1650
15	10	0.3	0.6	30	0.1600
16	10	0.3	0.7	10	0.1750
17	10	0.3	0.7	20	0.1700
18	10	0.3	0.7	30	0.1650
19	10	0.3	0.8	10	0.1800
20	10	0.3	0.8	20	0.1750
21	10	0.3	0.8	30	0.1700
22	10	0.3	0.9	10	0.1850
23	10	0.3	0.9	20	0.1800
24	10	0.3	0.9	30	0.1750
25	10	0.5	0.6	10	0.1900
26	10	0.5	0.6	20	0.1850
27	10	0.5	0.6	30	0.1800
28	10	0.5	0.7	10	0.1950
29	10	0.5	0.7	20	0.1900
30	10	0.5	0.7	30	0.1850

Configuración	Población	% Mutación	% Cruza	Generaciones	MAPE
31	10	0.5	0.8	10	0.2000
32	10	0.5	0.8	20	0.1950
33	10	0.5	0.8	30	0.1900
34	10	0.5	0.9	10	0.2050
35	10	0.5	0.9	20	0.2000
36	10	0.5	0.9	30	0.1950
37	10	0.7	0.6	10	0.2100
38	10	0.7	0.6	20	0.2050
39	10	0.7	0.6	30	0.2000
40	10	0.7	0.7	10	0.2150
41	10	0.7	0.7	20	0.2100
42	10	0.7	0.7	30	0.1350
43	10	0.7	0.8	10	0.1300
44	10	0.7	0.8	20	0.1550
45	10	0.7	0.8	30	0.1500
46	10	0.7	0.9	10	0.1550
47	10	0.7	0.9	20	0.1100
48	10	0.7	0.9	30	0.1150
49	20	0.1	0.6	10	0.1400
50	20	0.1	0.6	20	0.1350
51	20	0.1	0.6	30	0.1300
52	20	0.1	0.7	10	0.1450
53	20	0.1	0.7	20	0.1400
54	20	0.1	0.7	30	0.1380
55	20	0.1	0.8	10	0.1500
56	20	0.1	0.8	20	0.1450
57	20	0.1	0.8	30	0.1400
58	20	0.1	0.9	10	0.1550
59	20	0.1	0.9	20	0.1500
60	20	0.1	0.9	30	0.1450
61	20	0.3	0.6	10	0.1600
62	20	0.3	0.6	20	0.1550
63	20	0.3	0.6	30	0.1500
64	20	0.3	0.7	10	0.1650
65	20	0.3	0.7	20	0.1600
66	20	0.3	0.7	30	0.1550
67	20	0.3	0.8	10	0.1700

Configuración	Población	% Mutación	% Cruza	Generaciones	MAPE
68	20	0.3	0.8	20	0.1650
69	20	0.3	0.8	30	0.1600
70	20	0.3	0.9	10	0.1750
71	20	0.3	0.9	20	0.1700
72	20	0.3	0.9	30	0.1650
73	20	0.5	0.6	10	0.1800
74	20	0.5	0.6	20	0.1750
75	20	0.5	0.6	30	0.1700
76	20	0.5	0.7	10	0.1850
77	20	0.5	0.7	20	0.1800
78	20	0.5	0.7	30	0.1750
79	20	0.5	0.8	10	0.1900
80	20	0.5	0.8	20	0.1850
81	20	0.5	0.8	30	0.1800
82	20	0.5	0.9	10	0.1950
83	20	0.5	0.9	20	0.1900
84	20	0.5	0.9	30	0.1850
85	20	0.7	0.6	10	0.2000
86	20	0.7	0.6	20	0.1950
87	20	0.7	0.6	30	0.1900
88	20	0.7	0.7	10	0.2050
89	20	0.7	0.7	20	0.2000
90	20	0.7	0.7	30	0.1950
91	20	0.7	0.8	10	0.2100
92	20	0.7	0.8	20	0.2050
93	20	0.7	0.8	30	0.2000
94	20	0.7	0.9	10	0.1150
95	20	0.7	0.9	20	0.1100
96	**20**	**0.7**	**0.9**	**30**	**0.1050**
97	30	0.1	0.6	10	0.1300
98	30	0.1	0.6	20	0.1250
99	30	0.1	0.6	30	0.1200
100	30	0.1	0.7	10	0.1350
101	30	0.1	0.7	20	0.1300
102	30	0.1	0.7	30	0.1280
103	30	0.1	0.8	10	0.1400
104	30	0.1	0.8	20	0.1350

Configuración	Población	% Mutación	% Cruza	Generaciones	MAPE
105	30	0.1	0.8	30	0.1300
106	30	0.1	0.9	10	0.1450
107	30	0.1	0.9	20	0.1400
108	30	0.1	0.9	30	0.1350
109	30	0.3	0.6	10	0.1500
110	30	0.3	0.6	20	0.1450
111	30	0.3	0.6	30	0.1400
112	30	0.3	0.7	10	0.1550
113	30	0.3	0.7	20	0.1500
114	30	0.3	0.7	30	0.1450
115	30	0.3	0.8	10	0.1600
116	30	0.3	0.8	20	0.1550
117	30	0.3	0.8	30	0.1500
118	30	0.3	0.9	10	0.1650
119	30	0.3	0.9	20	0.1600
120	30	0.3	0.9	30	0.1550
121	30	0.5	0.6	10	0.1700
122	30	0.5	0.6	20	0.1650
123	30	0.5	0.6	30	0.1600
124	30	0.5	0.7	10	0.1750
125	30	0.5	0.7	20	0.1700
126	30	0.5	0.7	30	0.1650
127	30	0.5	0.8	10	0.1800
128	30	0.5	0.8	20	0.1750
129	30	0.5	0.8	30	0.1700
130	30	0.5	0.9	10	0.1850
131	30	0.5	0.9	20	0.1800
132	30	0.5	0.9	30	0.1150
133	30	0.7	0.6	10	0.1900
134	30	0.7	0.6	20	0.1850
135	30	0.7	0.6	30	0.1800
136	30	0.7	0.7	10	0.1950
137	30	0.7	0.7	20	0.1900
138	30	0.7	0.7	30	0.1850
139	30	0.7	0.8	10	0.2000
140	30	0.7	0.8	20	0.1950
141	30	0.7	0.8	30	0.1900

Configuración	Población	% Mutación	% Cruza	Generaciones	MAPE
142	30	0.7	0.9	10	0.1150
143	30	0.7	0.9	20	0.1100
144	30	0.7	0.9	30	0.1050

5.4. Resultados con imágenes en *escala de grises*

Con la idea de experimentar mediante un espacio de color más ligero computacionalmente, se ocuparon las imágenes en *escala de grises*, dándonos una idea de una posible configuración de parámetros para próximos experimentos.

Tomando en cuenta la configuración utilizada en los parámetros evolutivos en DeepGA para el procesamiento de imágenes en *escala de grises*, se observa que dos redes neuronales destacan de manera significativa en estas ejecuciones. El promedio de los resultados obtenidos, como se muestra en la tabla 8, revela un valor de 0.081 en MAPE. Este resultado sugiere una calidad destacable en la capacidad predictiva de estas redes neuronales. La elección de imágenes en *escala de grises* para la experimentación puede haber influido en la convergencia y desempeño de las redes neuronales. Al reducir la complejidad del espacio de color, es posible que las redes hayan logrado un aprendizaje más efectivo de los patrones relevantes en los datos. Por ende, es importante realizar una experimentación a color.

Estos experimentos fueron llevados acabo en una laptop DELL de 16 RAM, y una GPU Geforce RTX 3050, llevando un tiempo aproximado de 6 horas por ejecución. De igual manera, fue ocupado Google Colab Pro+ con las GPU T4 Y V100, con un tiempo promedio de 4 horas por ejecución.

+

Ejecución	**MAPE (Ecuación 2)**
1	0.065
2	0.065
3	0.088
4	0.067
5	0.085
6	0.069
7	0.078
8	0.102
9	0.086
10	0.104
Mejor	0.065
Peor	0.1043
Promedio	0.081
Desviación Estándar	0.107

Tabla 8: Resultados estadísticos de las ejecuciones con imágenes en *escala de grises*

En la Figura 20, se observa que la mayoría de las ejecuciones del algoritmo de optimización convergen hacia un valor bajo de MAPE, lo que indica una mejora significativa en la precisión del modelo a medida que aumenta el número de generaciones en el proceso evolutivo. Este comportamiento sugiere que el algoritmo está logrando refinar y ajustar las arquitecturas neuronales de manera efectiva con el tiempo, resultando en una reducción sistemática de los errores de predicción.

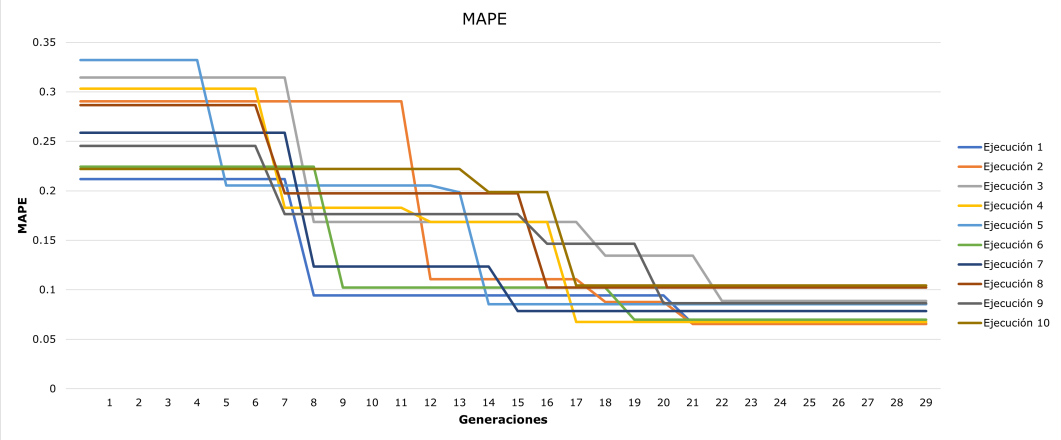

Figura 20: Gráfica de convergencia de 10 ejecuciones utilizando la función MAPE. La gráfica muestra cómo la métrica MAPE (Equación 2) evoluciona a lo largo de 30 generaciones en 10 ejecuciones diferentes.

En la Figura 21, se observa una alta variabilidad en el número de parámetros durante las primeras generaciones, con varias ejecuciones estabilizándose en configuraciones de parámetros más bajas hacia el final. Esta convergencia sugiere un proceso de optimización donde el algoritmo genético ajusta el número de parámetros para mejorar el rendimiento de las RNC.

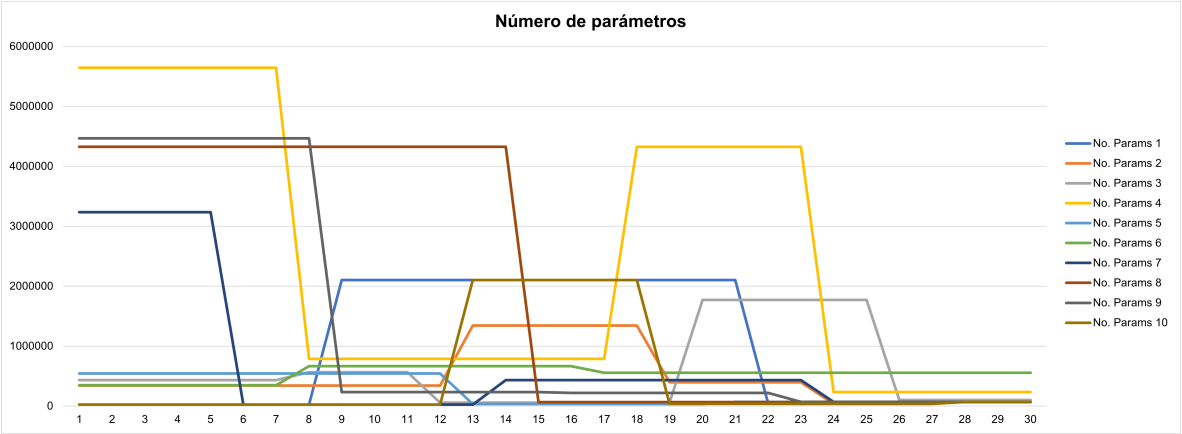

Figura 21: Gráfica en 10 ejecuciones de los parámetros de las RNC a lo largo de las generaciones del algoritmo genético. La gráfica muestra cómo el número de parámetros de las Redes Neuronales Convolucionales (RNC) evoluciona a través de 30 generaciones en 10 ejecuciones diferentes.

En la Figura 22, se observa que varias ejecuciones comienzan con un MSE alto, el cual disminuye progresivamente a medida que avanzan las generaciones. Sin embargo, algunas ejecuciones presentan fluctuaciones significativas en el MSE, indicando posibles variaciones en el proceso de optimización. En general, la mayoría de las ejecuciones tienden hacia un MSE más bajo, lo que sugiere una mejora en la precisión del modelo a lo largo del tiempo.

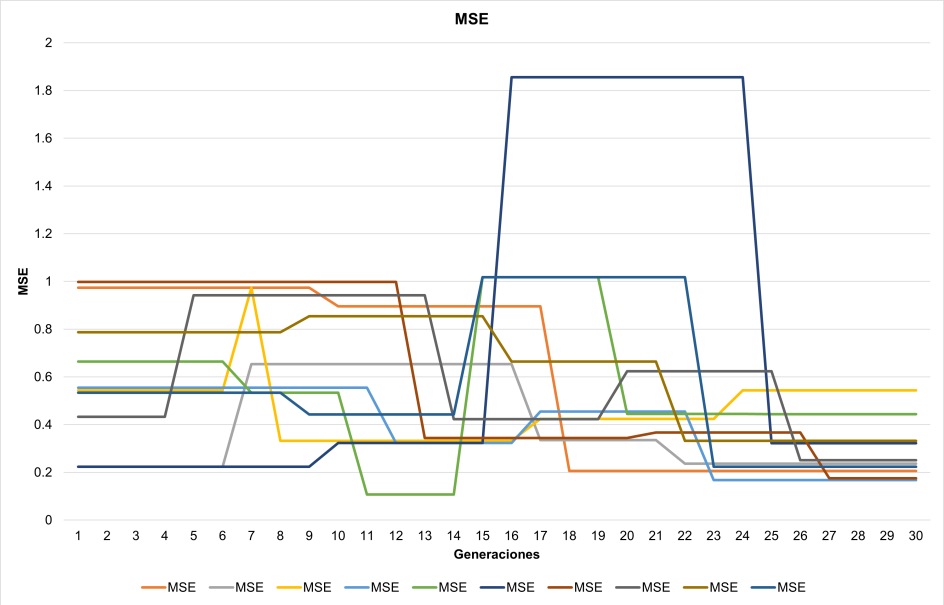

Figura 22: Gráfica de MSE (Ecuación 8) en 10 ejecuciones a lo largo de las generaciones del algoritmo genético. La gráfica muestra la evolución del MSE durante 30 generaciones en 10 ejecuciones distintas.

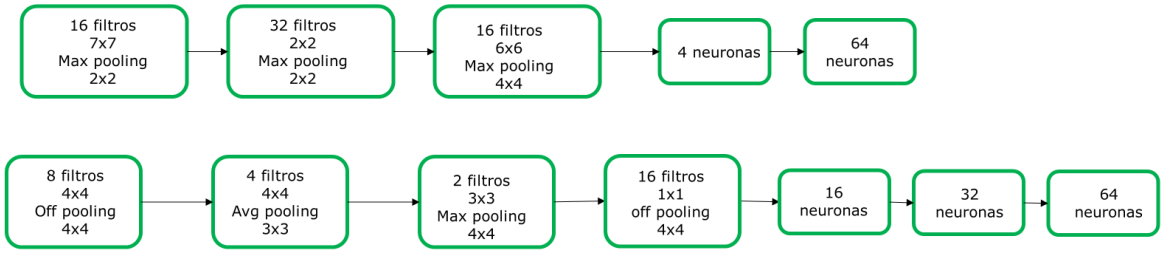

Figura 23: Arquitecturas de Redes Neuronales Convolucionales (RNC) encontradas en imágenes de escala de grises. La figura presenta dos propuestas de arquitecturas de RNC. La primera propuesta utiliza un filtro de 16 filtros de 7x7 en la primera capa, seguido de varias capas de pooling y convolución, y culmina con 4 neuronas en la primera capa totalmente conectada y 64 neuronas en la siguiente. La segunda propuesta comienza con 8 filtros de 4x4, incluye capas de off pooling y avg pooling, y termina con 16 neuronas en la primera capa totalmente conectada, aumentando a 32 y luego a 64 neuronas en las siguientes capas totalmente conectadas. Ambas arquitecturas están diseñadas para procesar eficientemente imágenes en escala de grises.

5.5. Resultados con imágenes a *color*

Utilizando los mismos parámetros y hiperparámetros de la tabla 4 para la inicialización de las RNC en la población inicial y los mismos parámetros en el algoritmo genético de la tabla 5, con la misma distribución de imágenes de las figuras 18, 19 se tiene este comportamiento en la convergencia ocupando imágenes a color. Como también, los parámetros de entrenamiento de la Tabla 6 en las RNC, con diferencia en el número de épocas, utilizando 10 épocas en estos experimentos. Los resultados de MAPE para las ejecuciones varían entre 0.0909 y 0.123. El mejor MAPE de 0.090 se obtuvo en la novena ejecución, indicando la menor tasa de error y, por lo tanto, la mejor precisión en esa instancia. El promedio de MAPE entre todas las ejecuciones es 0.108. Este valor promedio muestra que, en general, el modelo presenta una precisión bastante alta, aunque con algunas variaciones en el desempeño. Los resultados demuestran que el modelo tiene un buen desempeño en la estimación de ángulos de dirección con imágenes a color, con una precisión media alta y una baja variabilidad en el rendimiento. Estos experimentos fueron llevados acabo en una laptop DELL de 16 RAM, y una GPU Geforce RTX 3050, llevando un tiempo aproximado de 11 horas por ejecución. De igual manera, fue ocupado Google Colab Pro+ con las GPU T4 Y V100, con un tiempo promedio de 6 horas por ejecución.

Ejecución	**MAPE (Ecuación 2)**
1	0.098
2	0.112
3	0.112
4	0.123
5	0.112
6	0.097
7	0.097
8	0.122
9	0.090
10	0.113
Mejor	0.090
Peor	0.123
Promedio	0.108
Desviación Estándar	0.007

Tabla 9: Resultados estadísticos de las ejecuciones con imágenes a *color*

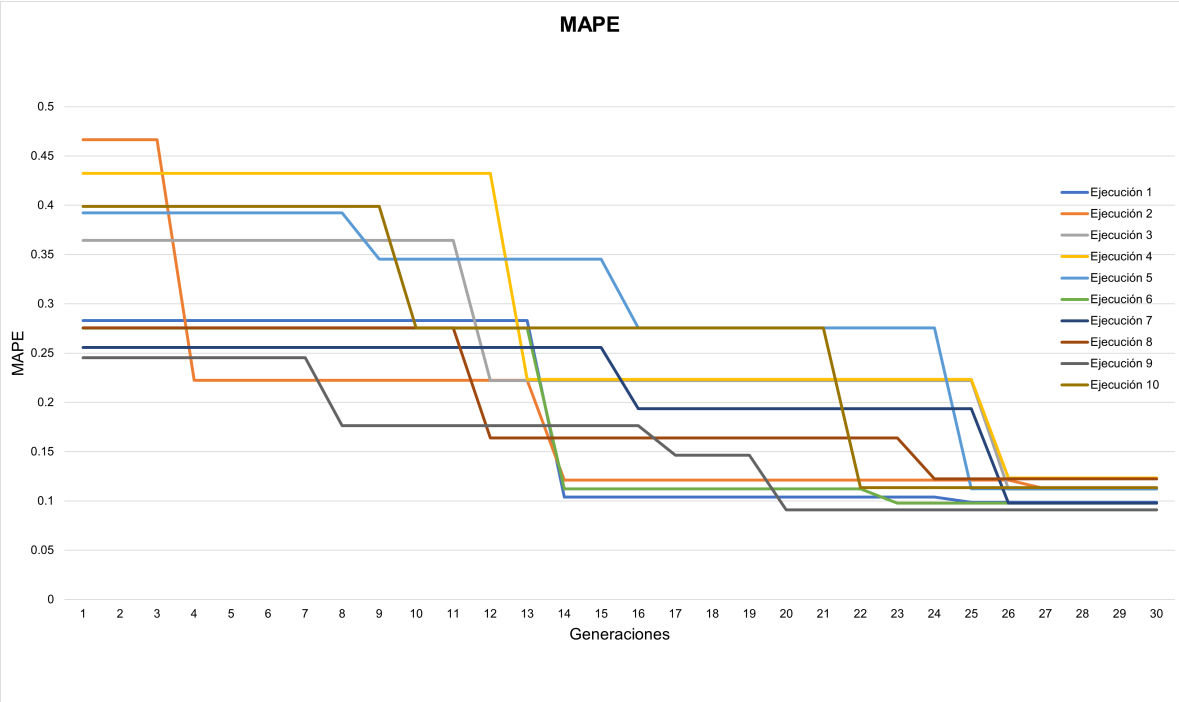

Figura 24: Gráfica de convergencia del MAPE (Equación 2) en 10 ejecuciones a lo largo de las generaciones del algoritmo genético.

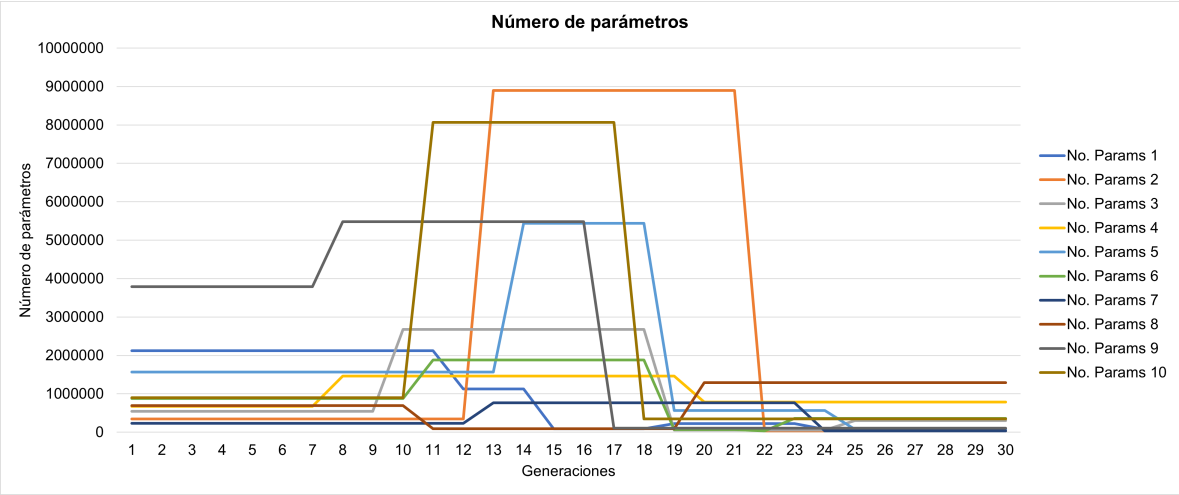

Figura 25: Gráfica en 10 ejecuciones de los parámetros de las RNC a lo largo de las generaciones del algoritmo genético con imágenes a color

Después de llevar a cabo 10 ejecuciones en el proceso de neuroevolución, se han identificado dos arquitecturas neurales destacadas que se presentan visualmente en la figura 27. Estas arquitecturas muestran una configuración consistente con 5 capas en total, compuestas por 3 capas convolucionales seguidas por 2 capas totalmente conectadas, teniendo 35,972 parámetros en su arquitectura.

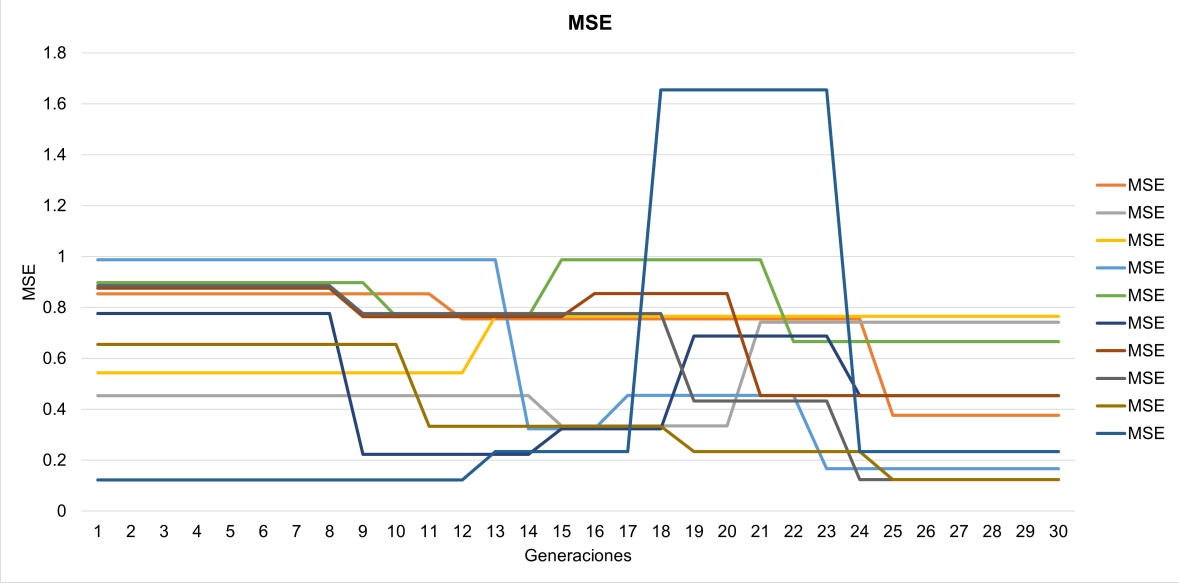

Figura 26: Gráfica de MSE (Equación 8) en 10 ejecuciones a lo largo de las generaciones del algoritmo genético.

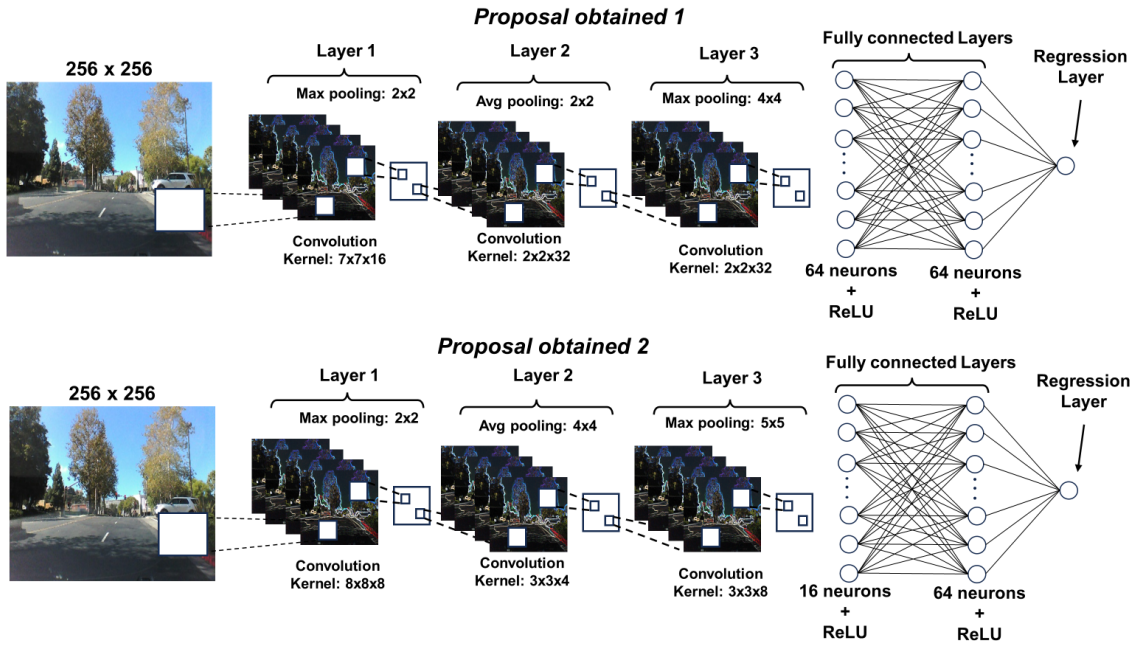

Figura 27: Arquitecturas de Redes Neuronales Convolucionales (RNC) encontradas en imágenes RGB. La figura compara dos propuestas de arquitectura de RNC. La primera propuesta utiliza un kernel de convolución de 7x7x16 en la primera capa, seguido de capas de pooling y convolución adicionales, culminando en capas totalmente conectadas con 64 neuronas. La segunda propuesta utiliza un kernel de 8x8x8 en la primera capa y presenta una estructura similar, pero con ajustes en los parámetros de pooling y convolución, así como una reducción en el número de neuronas en la primera capa completamente conectada a 16. Ambas arquitecturas terminan en una capa de regresión para generar la salida final.

5.6. Comparación de RNC

Con un entrenamiento de 10,000 imágenes en RGB del repositorio de [73], con la idea de poder establecer el número de parámetros de cada RNC, se realizó un comparativo con las arquitecturas para predicción de ángulos utilizando los hiperparametros de la tabla 10.
Estos experimentos fueron ocupado Google Colab Pro+ con las GPU T4 Y V100, con un tiempo promedio de 11 horas por ejecución.

Parámetro	**Valores**
Tamaño por lote	10
Épocas	50
Porcentaje de entrenamiento	0.0001
Número de imagenes de entrenamiento	8,000
Número de imagenes de prueba	2,000

Tabla 10: Hiperparámetros utiliados en entrenamiento de CNN

Para realizar una evaluación comparativa, consideraremos modelos previamente establecidos en la literatura y ampliamente utilizados en tareas similares. Estos modelos incluyen PilotNet [5], RestNet50 [8], VGG16, y VGG16-LSTM [7]. Adicionalmente, la segunda arquitectura propuesta y optimizada durante el proceso de optimización, con el enfoque PSO-RNC [18]. Este diseño experimental permitirá una comparación justa y detallada del rendimiento de las diferentes arquitecturas en términos de convergencia.

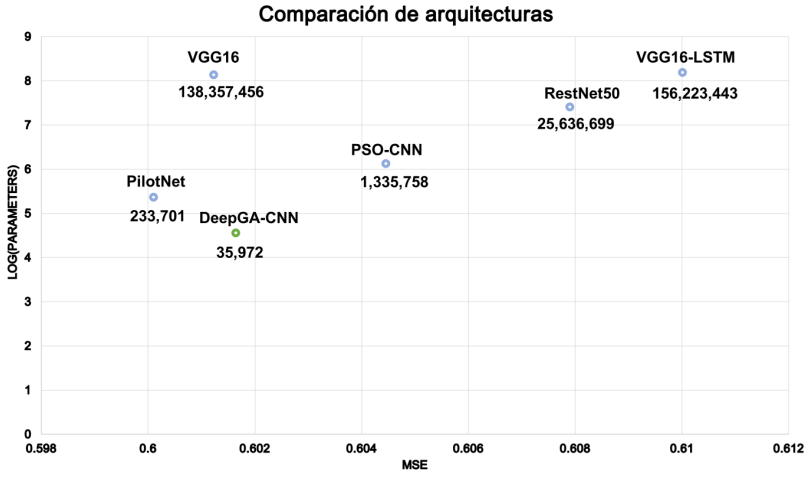

Figura 28: Comparación de arquitecturas de RNC en términos de MSE y el número de parámetros (en escala logarítmica). La gráfica muestra cómo diferentes arquitecturas de Redes Neuronales Convolucionales (RNC) se posicionan en función del MSE y la cantidad de parámetros que poseen.

En la Figura 28, se realiza una comparación detallada de diversas arquitecturas de redes neuronales convolucionales (RNC) a través de la distribución logarítmica de sus parámetros. El objetivo es visualizar y analizar las diferencias en complejidad entre estas arquitecturas. Cada valor de parámetro se transforma utilizando la función log_{10} para distribuir la escala y resaltar las variaciones. Las arquitecturas evaluadas incluyen PilotNet [5] con 233,701 parámetros, RestNet50 [8] con 25,636699, PSO-RNC [18] con 1,335,758, VGG16 con 138,357456, VGG16-LSTM [7] con 156,223,443, y la propuesta DeepGA-RNC (de la figura 27) con 35,972 parámetros. La escala logarítmica revela una apreciación más equitativa de las diferencias en complejidad, destacando la eficiencia de DeepGA-RNC con una cantidad inferior de parámetros. A pesar de la diversidad en la cantidad absoluta de parámetros, cada arquitectura fue entrenada obteniendo un MSE en un rango entre 0.602121 y 0.6123242, esto dado a la cantidad de imágenes utilizadas.

5.7. Estadísticas RNC

Utilizando las RNC mostradas en la figura 28 se realizan 10 ejecuciones, donde en cada MSE se ocupa el mismo conjunto de datos con un 80 % entrenamiento y 20 % prueba, utilizando los hiperparámetros de la tabla 10. Cada ejecución implica el entrenamiento de las RNC en un conjunto de datos utilizando condiciones específicas, como 50 épocas de entrenamiento, un tamaño de lote (batch) de 10 muestras por iteración y un porcentaje de aprendizaje de 0.0001. Al realizar 10 ejecuciones independientes, se obtienen 10 conjuntos de valores de MSE, uno para cada ejecución. Estos valores se registran para analizar la consistencia y variabilidad en el rendimiento de las RNC bajo las condiciones experimentales dadas.

Parameter	**DeepGA-CNN**	**PSO-CNN**	**VGG16**	**VGG16-LSTM**	**RestNet-50**	**PilotNet**
MSE 1	0.601643	0.604453	0.601226	0.601233	0.607896	0.601643
MSE 2	0.602232	0.605423	0.601343	0.609756	0.605433	0.600887
MSE 3	0.603212	0.605332	0.601232	0.609754	0.605433	0.600976
MSE 4	0.601112	0.604232	0.601333	0.61238	0.608757	0.601979
MSE 5	0.601234	0.603434	0.601432	0.610975	0.607764	0.600892
MSE 6	0.601432	0.604232	0.601322	0.610086	0.607654	0.600123
MSE 7	0.601212	0.605343	0.601645	0.608754	0.608765	0.600166
MSE 8	0.600943	0.606534	0.601223	0.608754	0.606878	0.600687
MSE 9	0.601453	0.604328	0.601323	0.604543	0.606775	0.600855
MSE 10	0.601123	0.605433	0.601234	0.612354	0.607648	0.600954

Tabla 11: Resultados de las métricas MSE para cada modelo

Posteriormente, se llevó a cabo una prueba de $Kolmogorov - Smirnov$ para evaluar la normalidad de los datos de MSE de cada modelo. Los resultados indican un valor bajo para el estadístico de prueba $D = 0.725$ y un valor $p = 1.084 \times 10^{-32}$, lo cual es menor a $\alpha = 0.05$. Esto lleva al rechazo de la hipótesis nula, sugiriendo que los datos de MSE no provienen de una distribución normal. En consecuencia, se descarta la no normalidad de los datos de MSE, lo cual debe considerarse al seleccionar métodos estadísticos.
Considerando esto, se realiza la prueba de $Kruskal - Wallis$ lo que nos revela resultados significativos, con un estadístico de prueba $D = 47.319$ y un valor $p = 4.889 \times 10^{-9}$,lo cual es menor a $\alpha = 0.05$. Esto conduce al rechazo de la hipótesis nula, indicando que al menos una mediana es diferente en los grupos analizados. Por lo tanto, se infiere la presencia de diferencias estadísticamente significativas en las medianas entre los grupos evaluados.

Para analizar las diferencias entre el modelo DeepGA-RNC y otros modelos, se aplicó la prueba de $Mann - WhitneyU$, cuyos resultados se presentan en la tabla 12. La finalidad de

Grupo	DeepGA-CNN
DeepGA-CNN	-
PSO-CNN	0.000181651
VGG16	1
VGG16-LSTM	0.001007976
RestNet-50	0.000182672
PilotNet	0.015525522

Tabla 12: Resultados de las comparaciones de cada modelo con la prueba de $Mann - WhitneyU$

esta prueba fue establecer si las diferencias observadas eran estadísticamente significativas, es decir, si eran mayores o menores que el nivel de significancia $\alpha = 0.05$ Los valores en esta tabla indican comparaciones de rendimiento entre DeepGA-RNC y diferentes modelos, señalando discrepancias en criterios específicos. No se identifican diferencias significativas entre DeepGA-RNC y PSO-RNC, VGG16, y VGG16-LSTM, ya que las diferencias son mínimas. En cambio, se observa una diferencia mayor con PilotNet, donde DeepGA-RNC exhibe un valor aproximadamente 0.0155 superior. Esto sugiere que podría existir una divergencia significativa en el rendimiento entre estos dos modelos. En contraste, las variaciones con RestNet-50 son las no significantes. Por lo tanto, se puede decir que DeepGA-RNC presenta un comportamiento similar a PSO-RNC, VGG16, y VGG16-LSTM, a pesar de tener un menor número de parámetros. Sin embargo, se observa una diferencia con RestNet50 y PilotNet, ambos con un mayor número de parámetros en comparación con DeepGA-RNC.

5.8. Latencia de RNC

Las redes más profundas pueden capturar patrones más detallados y abstractos en los datos, lo que generalmente resulta en un mejor rendimiento. Sin embargo, aumentar la profundidad de la red también tiene un costo: el incremento en el tiempo de procesamiento, conocido como latencia. Mientras que redes más profundas pueden ofrecer mejores capacidades de aprendizaje y mayor precisión, también introducen una mayor latencia, lo que puede ser un impedimento en aplicaciones que requieren respuestas rápidas. Por ejemplo, PilotNet y DeepGA-RNC presentan las menores latencias, siendo adecuadas para aplicaciones en tiempo real como los vehículos autónomos. En contraste, PSO-RNC muestra la mayor latencia, lo que puede ser menos práctico en situaciones que requieren toma de decisiones inmediatas.

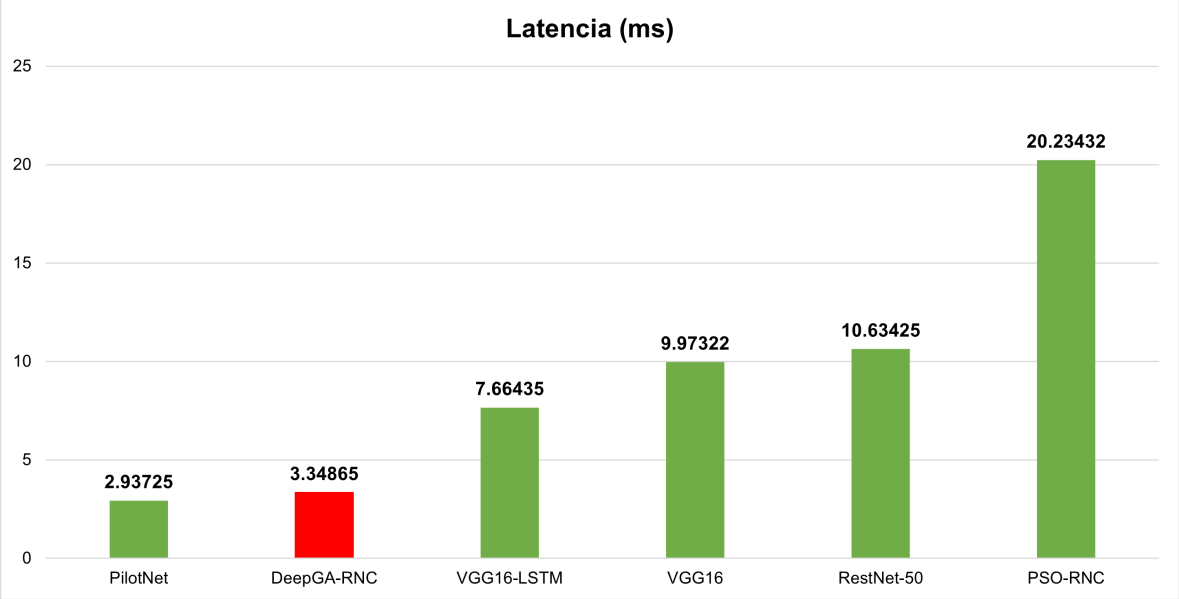

Figura 29: Comparación de la latencia (en milisegundos) después de ser entrenadas diferentes arquitecturas de Redes Neuronales Convolucionales (RNC), midiendo el tiempo de reacción por imagen. Se observa que PilotNet presenta la menor latencia, mientras que PSO-RNC muestra la mayor latencia, indicando una variabilidad significativa en el tiempo de procesamiento entre diferentes modelos

Por lo tanto, es importante encontrar un equilibrio adecuado entre la complejidad del modelo y la eficiencia del procesamiento para cumplir con los requisitos de rendimiento específicos de cada aplicación. Este balance es importante en campos como los vehículos autónomos, donde una rápida toma de decisiones es vital para la seguridad y la eficacia del sistema. Al diseñar o seleccionar una RNC, se debe considerar tanto la capacidad de aprendizaje necesaria como las restricciones de latencia para asegurar el rendimiento óptimo del sistema en su contexto de aplicación.

5.9. Latencia y RNC en sistemas embebidos

La combinación de una RNC de baja latencia con un dispositivo embebido adecuado puede maximizar la eficiencia y la velocidad de procesamiento [77]. Para abordar estas exigencias, es importante considerar los dispositivos que, por su arquitectura y capacidad, están especialmente diseñados para manejar modelos de aprendizaje profundo en entornos con restricciones de recursos. Entre los sistemas embebidos más destacados y que mejor se adaptan a estas condiciones se encuentran:

Dispositivo	**Procesador**	**Consumo Energético**	**Interfaces**	**Aplicaciones Principales**
NVIDIA Jetson Xavier NX	6 núcleos ARM, GPU Volta, 8 GB LPDDR4	10-15 W	1x M.2 Key E, 1x M.2 Key M	Robótica, Visión Artificial
Google Coral Dev Board	Quad-core ARM Cortex-A53, Edge TPU	5 W	1x USB 3.0, 1x GPIO, 1x PCIe	Edge AI, Procesamiento de Imágenes
Intel Movidius Neural Compute Stick 2	Myriad X VPU	1 W	USB 3.0	Inferencia de Redes Neuronales
Raspberry Pi 4 + Google Coral USB Accelerator	Quad-core ARM Cortex-A72 + Edge TPU	Variable (Pi 4: 5-7 W, Coral: 2 W)	USB 3.0, GPIO, HDMI	Prototipos, Edge AI
Qualcomm Snapdragon NPE	Hexagon DSP, Adreno GPU	Variable (móvil)	USB, GPIO, PCIe	Aplicaciones Móviles, AI
Xilinx ZCU102 Evaluation Kit	Zynq UltraScale+ MPSoC	Variable (configurable)	HDMI, PCIe, Ethernet	Aceleración de IA, FPGA

Tabla 13: Comparación de dispositivos para el procesamiento de RNCs

La tabla 13 muestra dispositivos avanzados para el procesamiento de RNC, enfocados en aplicaciones que requieren baja latencia. Entre ellos, el NVIDIA Jetson Xavier NX es reconocido por su alta capacidad de procesamiento y flexibilidad, lo que lo convierte en una elección preferida para robótica y visión artificial. Estudios han demostrado su efectividad en aplicaciones de visión computacional en tiempo real debido a su capacidad de balancear rendimiento y consumo energético [78].

El Google Coral Dev Board destaca en el ámbito de la IA y el procesamiento de imágenes. Su bajo consumo energético lo convierte en una opción óptima para dispositivos autónomos que requieren eficiencia en la ejecución de modelos de IA sin comprometer el rendimiento [79]. Por otro lado, el Intel Movidius Neural Compute Stick 2 es importante para tareas de inferencia rápida, especialmente en dispositivos con recursos limitados. Su arquitectura Myriad X VPU es capaz de realizar inferencias de RNC con un consumo mínimo de energía, manteniendo la latencia baja, lo que lo hace ideal para aplicaciones embebidas y móviles [80]. La combinación de Raspberry Pi 4 y Google Coral USB Accelerator permite el desarrollo de prototipos flexibles de Edge AI, balanceando rendimiento y consumo energético.

Este conjunto es ampliamente utilizado en la creación de soluciones de bajo costo y alta eficiencia para aplicaciones de inteligencia artificial [81]. El Qualcomm Snapdragon NPE está optimizado para aplicaciones móviles, combinando un DSP Hexagon y una GPU Adreno para ofrecer procesamiento en tiempo real con un consumo energético adaptable, esencial para dispositivos portátiles que requieren un rendimiento continuo y eficiente [82] . Finalmente, el Xilinx ZCU102 Evaluation Kit, basado en FPGA, ofrece una flexibilidad única para la aceleración de IA, permitiendo configuraciones personalizadas que maximizan la eficiencia de procesamiento para aplicaciones críticas donde la latencia ultra baja es imprescindible [83]. Estos dispositivos pueden ser los más útilies para el procesamiento de las RNC con la latencia de la imagen 29.

Capítulo 6
Conclusiones y trabajos futuros

Los resultados obtenidos en esta investigación destacan la efectividad y el impacto del uso de algoritmos genéticos en la optimización de RNC para la predicción de ángulos de dirección en vehículos autónomos. Específicamente, los parámetros del algoritmo genético han demostrado una convergencia exitosa, lo que refleja una exploración dinámica dentro del espacio de soluciones. Esta exploración ha sido guiada de manera efectiva por la diversidad generada en la población y la adecuada parametrización del algoritmo, lo que ha permitido la identificación de arquitecturas óptimas para la tarea propuesta.

Los parámetros sugiere que el equilibrio entre la exploración y la explotación ha sido adecuadamente manejado, evitando problemas como la convergencia prematura en soluciones subóptimas. La diversidad dentro de la población ha jugado un papel crucial en este proceso, asegurando que se exploren múltiples regiones del espacio de soluciones antes de que el algoritmo se concentre en las áreas más prometedoras.

Además, la investigación ha demostrado que el enfoque de neuroevolución no solo es eficaz, sino que también presenta un gran potencial para mejorar la precisión en la predicción de ángulos de dirección en vehículos autónomos. La capacidad de la neuroevolución para adaptar la arquitectura de la red a las necesidades específicas de la tarea subraya su ventaja frente a métodos tradicionales, donde las arquitecturas de las redes neuronales suelen diseñarse manualmente y sin la flexibilidad necesaria para responder a diferentes escenarios o requerimientos.

Un hallazgo clave de este trabajo es la posibilidad de reducir significativamente la latencia en el procesamiento de RNCs mediante la aplicación de neuroevolución. La capacidad de generar diseños específicos de RNC optimizados para baja latencia permite obtener una respuesta mucho más rápida, lo que es crucial en aplicaciones en tiempo real como la conducción autónoma. La reducción de la latencia no solo mejora el rendimiento del sistema en términos de velocidad de respuesta, sino que también puede aumentar la seguridad y la fiabilidad del vehículo al permitir decisiones más rápidas y precisas en situaciones dinámicas.

Este trabajo sugiere que la combinación de neuroevolución y RNCs representa un avance significativo hacia la creación de sistemas de conducción autónoma más eficientes y adaptativos. Al optimizar tanto la estructura de la red como sus parámetros de funcionamiento, se puede obtener un sistema que no solo es más preciso en la predicción de ángulos de dirección, sino que también responde de manera más eficiente a los requerimientos del entorno, reduciendo la latencia y mejorando la experiencia de conducción autónoma. Estos hallazgos abren nuevas vías para futuras investigaciones, donde se podría explorar la integración de neuroevolución

con otras técnicas de optimización y su aplicación en diferentes aspectos del control autónomo. Es fundamental que, aunque estos resultados son prometedores, no deben considerarse como resultados ni conclusiones definitivas. Se proyecta continuar trabajando en experimentaciones con DeepGA, explorando conjuntos de datos adicionales, como el conjunto de datos de Udacity [14], que presenta una distribución de ángulos de dirección diferente. Esto permitirá evaluar la robustez y generalización de las arquitecturas en contextos más diversos de conducción autónoma.

A lo largo de los experimentos, se observa una variabilidad en el número de parámetros y MSE durante el proceso evolutivo del algoritmo genético. Esta variación es totalmente esperada debido a la diversidad en las diferentes estructuras de los individuos en cada generación. Sin embargo, se reconoce la necesidad de un análisis más profundo y la consideración de funciones de aptitud adicionales para la construcción de estas redes. Esto podría contribuir a lograr una mayor consistencia y un rendimiento más estable a lo largo del entrenamiento.

En este sentido, se está trabajando en el desarrollo de una propuesta de función multiobjetivo para futuras experimentaciones en DeepGA. Esta función multiobjetivo busca abordar la complejidad del problema de optimización, incorporando múltiples criterios de rendimiento para guiar la evolución hacia soluciones más equilibradas y eficientes con la idea de potenciar aún más la capacidad de DeepGA en el diseño de RNC para conducción autónoma. Para darle continuidad a la investigación, se tiene previsto ampliar y enriquecer tanto el marco teórico como la explicación detallada de la metodología utilizada. En relación al marco teórico, se buscará profundizar en las bases teóricas de la neuroevolución y la construcción de redes neuronales, incorporando conceptos avanzados y enfoques novedosos que puedan enriquecer la comprensión del lector. Esto incluirá una revisión exhaustiva de las tendencias actuales en el campo de la optimización de arquitecturas de redes neuronales y la neuroevolución aplicada a problemas específicos, con el objetivo de contextualizar aún más los resultados obtenidos. Adicionalmente, se pretende proporcionar una explicación más detallada de la metodología empleada en los experimentos. Esto implica una descripción más profunda de los parámetros y configuraciones específicas utilizadas en DeepGA, así como una justificación más completa de las decisiones tomadas en cada etapa del proceso experimental.

Anexos

Una de las contribuciones más destacadas de esta tesis fue la presentación del artículo científico en el MCPR 2024, como titulo, *Reducing Parameters by Neuroevolution in CNN for Steering Angle Estimation* [84].

Reducing Parameters by Neuroevolution in CNN for Steering Angle Estimation

José-David Velazco-Muñoz( ), Héctor-Gabriel Acosta-Mesa, and Efrén Mezura-Montes

Artificial Intelligence Research Institute, University of Veracruz, Veracruz 91097, Mexico
zs22000515@estudiantes.uv.mx, {heacosta,emezura}@uv.mx

Abstract. Convolutional Neural Networks (CNNs) are becoming increasingly popular in autonomous driving. Researchers are focused on optimizing these models to work on smaller control devices, which can allow for more performance of vehicle steering. A lot of attention is being paid to reducing the number of parameters in a CNN so that it can predict steering angles more efficiently for autonomous vehicles. To tackle this challenge, we have adopted a neuroevolution approach, which is a fusion of neural networks and evolutionary algorithms. This approach can help us find models that are less complex but still have adequate predictive capacity and a lower number of parameters compared to existing networks. The results show that the neuroevolution approach was successful in producing an architecture with only 35,972 evaluated parameters. This is a significant reduction in the number of parameters compared to established models like PilotNet with 233,701 parameters, ResNet50 with 25,636,699 parameters, PSO-CNN with 1,335,758 parameters, VGG16 with 138,357,456 parameters, and VGG16-LSTM with 156,223,443 parameters. Despite having fewer parameters, our model's performance is similar to some of the other CNNs.

Keywords: Steering angle · neuroevolution · CNN · autonomous driving car

1 Introduction

The development of Convolutional Neural Networks (CNNs) has had a significant impact on autonomous driving systems in recent years. These networks have the ability to process complex visual information and have become a critical technology for vehicular autonomy [6]. The incorporation of CNNs in self-driving car control systems has the potential to revolutionize the automotive sector and urban transportation. These systems rely on the steering angle to control the trajectory of the vehicle. By enabling precise control, the vehicle can safely and efficiently follow its intended path, steer clear of accidents, and stay in the correct lane. Designing a CNN can be a challenging task. Determining its parameters

E. Mezura-Montes et al. (Eds.): MCPR 2024, LNCS 14755, pp. 377–386, 2024.
https://doi.org/10.1007/978-3-031-62836-8_35

is equally important as it significantly influences the performance of the results and convergence. The main parameters and hyperparameters of a CNN include the learning rate, the number of epochs, the batch size, the activation function, the number and sequence of layers (convolutional, pooling, and fully connected), the number of neurons in the fully connected layers, and the number and size of filters in each convolutional layer. All these parameters play an essential role in ensuring the correct performance of the network. Addressing these challenges and optimizing fundamental aspects within a network could open up new possibilities in architecture design. Integrating neural networks with evolutionary algorithms through neuroevolution presents a novel approach to enhancing neural networks [5,14].

This paper is divided into four main sections. Section 2, related works, explores previous research related to the study topic. Section 3, methods and materials, explains the process used for conducting this study, including the dataset and algorithm used. Section 4 presents the experiments conducted and a discussion of each of them. Finally, Sect. 5 provides conclusions and a discussion of the overall findings.

2 Related Works

In the field of steering angle prediction, there are two different research paths [12]. The first involves using established vision methods to process images and identify road boundaries or lane markings, and then calculating the steering angle based on this information. The second path involves using CNNs for imitation learning to predict steering angles in both real-world scenarios and simulated applications. Various CNNs have been explored for this purpose. Several CNNs have been developed and tested to improve the accuracy and performance of steering angle predictions for autonomous vehicles. One well-known network is PilotNet [1], which is a deep architecture that directly maps camera images in a vehicle to steering angles. This eliminates the need for manual feature design, making it more efficient. In a study presented in [4], an autonomous car prototype and CNNs on Raspberry Pi were used. Another proposal, [17], is based on CNN-LSTM to predict steering angles. This proposal uses future sequential information during training, which results in good performance with the Udacity dataset [15] and in real autonomous vehicle tests. Similarly, in [7,8], proposals and experiments with CNN using simulated data are presented. Researchers have also explored optimizing algorithms and employing transfer learning techniques with CNNs to enhance their capability in predicting steering angles. The study [9] evaluated the performance of pre-trained networks, including AlexNet, ResNet18, and DenseNet121, by applying transfer learning and adjusting the last layer. ResNet18 and DenseNet121 turned out to have a better error percentage than the others. Among the various optimization methods used in CNNs, there are compression and acceleration techniques [3]. These techniques include pruning, factorization, and knowledge distillation which reduce the model's depth without compromising its performance. When it comes to optimizing CNNs

using evolutionary computation, there are various approaches available. One such approach is to use memetic algorithms, which refine the mutation step through a series of educated local-search moves. A study presented in [10] discusses this paradigm. Another approach is to use Particle Swarm Optimization (PSO), which was used in [13] to optimize the weights of a CNN for angle estimation in autonomous driving. The study compared PSO and bat algorithms, and demonstrated the quality of the predictions through Mean Squared Error (MSE). The Udacity dataset [15] was used to validate different sets of hyperparameters. Existing techniques for compressing CNNs have shown evidence of increased computational efficiency and improved generalization, particularly in pruning techniques. However, not all techniques work well for certain problems and may require fine-tuning. On the other hand, methods like particle swarm optimization (PSO), memetic algorithms, and transfer learning have shown to generate higher precision CNNs, but with greater depth in the architectures. A literature review summary indicates that hyperparameter optimization, transfer learning, and compression techniques are the main focus of research in this area. These methods have demonstrated positive results in improving model performance. However, there is a lack of detailed exploration of the internal parameters and network structure depending on the problem. This lack of attention highlights the need for further research, leading to the adoption of neuroevolution techniques, which are more adaptable compared to other techniques. It is essential to understand the impact of the quantity of parameters on the latency of control devices, emphasizing the importance of further research.

3 Materials and Methods

This methodology, illustrated in Fig. 1, compares various CNNs for estimating steering angles in autonomous driving. It distinguishes itself by considering the number of parameters used by these networks. The main objective is to examine how varying parameter counts in CNN architectures impact the accuracy

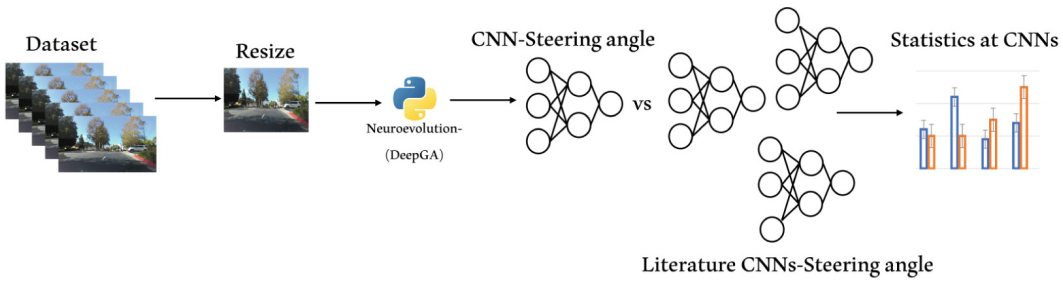

Fig. 1. This proposed methodology employs a dataset resized to 256×256, which is then fed into a neuroevolution algorithm to obtain a CNN architecture. This architecture is subsequently compared with the most relevant CNNs in the steering angle field. Finally, statistical comparisons are made between the CNNs and the proposed CNN to identify any significant differences.

of direction estimation while performing autonomous vehicle operations. This study emphasizes the significance of restricting the number of parameters used in architectures designed for predicting steering angles.

3.1 Dataset

This research proposes using the PilotNet dataset [2], as shown in Fig. 2, which is a representative dataset in autonomous vehicle research for predicting angles using only 1000 RGB images. The images were resized from 255×455 pixels to 255×255 pixels to accommodate the necessary features input of DeepGA. Out of all the images, 70% were used for model training, while the remaining 30% were used for testing during the neuroevolution process. Additionally, 10000 more images were used under the same conditions for model comparison.

Fig. 2. Images labeled with the required steering angle of the vehicle's steering wheel, related to the specific characteristics of the road observable in the image. In these images, the steering angle is associated with particular elements of the road environment, such as curves, lane changes, U-turns, among other factors that affect the vehicle's direction.

3.2 Neuroevolution

We utilized a neuroevolution algorithm to generate our network, specifically the Deep Genetic Algorithm (DeepGA) [16]. DeepGA is an evolutionary algorithm designed to optimize CNNs for visual recognition tasks, such as classifying. It employs a hybrid encoding to represent complex neural network structures and automatically improves the architecture through genetic algorithms instead of manual design. Unlike other neuroevolution methods [11], optimization and compression techniques, DeepGA has great adaptability and few tuning parameters. For the design of our network, primarily focused on regression CNNs, modifications were made to the DeepGA algorithm. The algorithm was adapted to generate designs for regression networks by altering the maximization criteria to minimization. The objective was to discover networks with the lowest values in

Mean Absolute Percentage Error (MAPE) (Eq. 1) which is a measure that evaluates the accuracy of a model by calculating the average percentage difference between predicted values and actual values, relative to the actual values.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{1}$$

We modified the algorithm to find networks that are both accurate in their predictions and efficient in their use of resources. This means we aim to obtain models that are compact and effective. To introduce more diversity, we use a variant of elitism, which considers the greater among the lesser as a means of generating diversity. Our objective is to discover efficient networks that use fewer parameters but still deliver high predictive performance.

4 Experiments and Results

Using the first 1000 images from the dataset [2], neuroevolution was performed. The initial ranges used by DeepGA to evolve CNNs (minimum and maximum counts for convolutional and fully connected layers, number of filters, filter dimensions, pooling methods, pool sizes, and neuron quantities) are summarized in Table 1. CNN models were trained using the parameters and hyperparameters specified in Table 1, they were taken concerning [16], given the promising results observed in the internal constructions of CNNs in classification. Those DeepGA parameters in Fig. 2 were fine-tuned by a grid search process. After conducting

Table 1. Initialization parameters and hyperparameters in CNNs

Parameter	Values
Minimum Convolutional Layers	2
Maximum Convolutional Layers	6
Minimum Fully Connected Layers	2
Maximum Fully Connected Layers	4
Number of Filters	2, 4, 8, 16, 32
Filter Size	2, 3, 4, 5, 6, 7, 8
Pooling Type	Max, Avg
Pooling Size	2, 3, 4, 5
Number of Neurons	4, 8, 16, 32, 64, 128
Training Epochs	30
Learning Rate (Adam Optimizer)	1×10^{-4}
Batch Size	10

Table 2. DeepGA Parameters

Parameter	Values
Population Size	20
Number of Generations	30
Tournament Size	5
Crossover Percentage	0.9
Mutation Percentage	0.7

ten DeepGA independent runs on a Tesla V100 in Google Colab with 16 GB with 13 h to complete one run, we obtained the following statistical results: the best value was 0.090, the worst value was 0.123, the mean value was 0.108, and the standard deviation was 0.007. Such values indicate a competitive and robust performance on DeepGA based on the fitness values reached. Moreover, the run located in the median value of the ten runs is plotted in Fig. 3. The MAPE convergence plot on the top left side of Fig. 3 shows that DeepGA improves result quality by avoiding local optimum solutions. Similarly, on the top right side of Fig. 3, the behavior of MSE exhibits a similar trend to MAPE. This leads us to consider that the fitness function is more representative in terms of information regarding prediction quality in CNN construction within DeepGA. After exploring different configurations in CNN parameters, DeepGA finds more compact designs without compromising regression quality across generations, as shown in the bottom part.

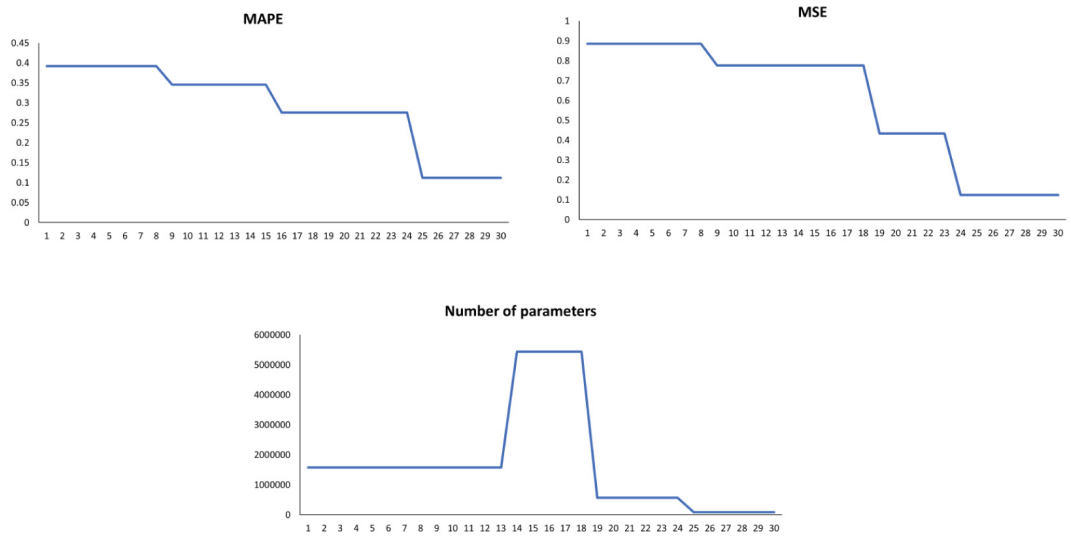

Fig. 3. Convergence plot of the run located at the median value of 10 executions across 30 generations of the fitness function (MAPE), MSE, and the number of parameters.

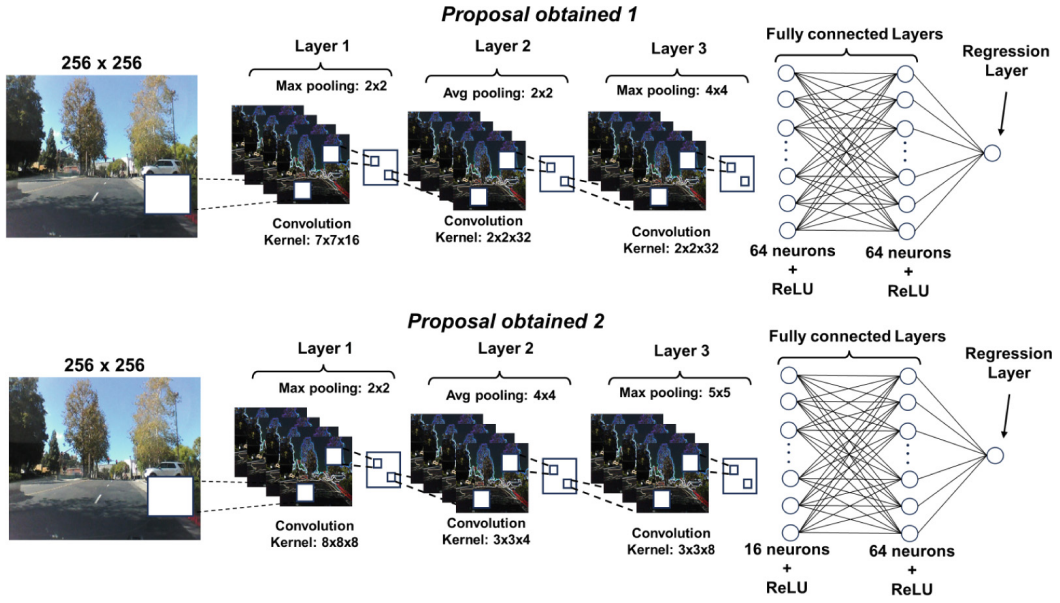

Fig. 4. Best architectures obtained by DeepGA. In the first proposal, the architecture consists of three convolutional layers using convolution kernels of sizes 7×7, 2×2, with pooling kernel sizes of 2×2 and 4×4. Additionally, there are two fully connected layers, each with 64 neurons. In the second proposal, the architecture includes three convolutional layers using convolution kernels of sizes 8×8, 3×3, with pooling kernel sizes of 2×2, 4×4, and 5×5. Similarly, there are two fully connected layers, one with 16 neurons and the other with 64 neurons.

The details of the best two architectures are depicted in Fig. 4. For comparison purposes with those state-of-the-art approaches, the first one was chosen. The models used for comparison include PilotNet [1], RestNet50 [8], VGG16, and VGG16-LSTM [17], and the optimized by PSO in [13]. All compared architectures will undergo an identical training process, using a batch size of 10 for 50 epochs and a fixed learning rate of 0.0001. This experimental design will ensure a fair and comprehensive comparison of the learning performance of different architectures.

In Fig. 5, a detailed comparison of those considered CNN architectures is conducted through the logarithmic distribution of their parameters. The aim is to visualize and analyze the differences in complexity among those architectures. Each parameter value is transformed using the log_{10} function to distribute the scale and highlight variations. Those evaluated architectures include PilotNet [1] with 233,701 parameters, RestNet50 [8] with 25,636,699, PSO-CNN [13] with 1,335,758, VGG16 with 138,357,456, VGG16-LSTM [17] with 156,223,443, and the proposed DeepGA-CNN (from Fig. 4) with 35,972 parameters (see Table 3). Considering the significant differences in the number of parameters required by the compared networks, the parameter axis is represented by its logarithm value for better visualization, emphasizing the performance of DeepGA-CNN with a low number of parameters. Despite the diversity in the absolute number of parameters, each architecture was trained, achieving a MSE in a range

Table 3. Comparison of Convolutional Neural Network Models

Model	Parameters	MSE
DeepGA-CNN	35,972	0.601
PSO-CNN	1,335,758	0.604
VGG16	138,357,456	0.601
PilotNet	233,701	0.600
RestNet-50	25,636,699	0.607
VGG16-LSTM	156,223,443	0.610

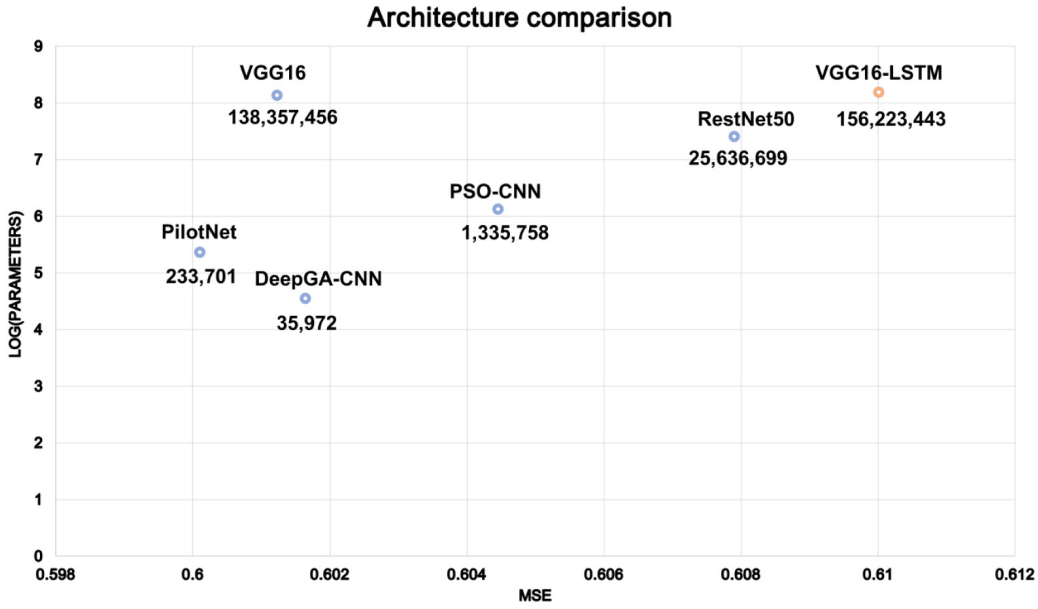

Fig. 5. Architecture distribution through its MSE and number of parameters

between 0.602 and 0.612, attributed to the number of images used. Ten experiments were conducted for each CNN. We conducted 10 runs to ensure a more solid and comprehensive sampling, encompassing different partitions and thus ensuring a more thorough and reliable evaluation of the results, divided into 70% for training and 30% for testing Training the CNNs involved specific conditions for each run, including 50 training epochs, a batch size of 10 samples per iteration, and a learning rate of 0.0001, 10,000 images. After the ten independent runs, ten sets of MSE values were obtained. A $Kolmogorov - Smirnov$ test indicated non-normality in the MSE data samples, and a $Kruskal - Wallis$ test revealed statistically significant differences in medians among the evaluated groups. To analyze differences between the DeepGA-RNC model and others, the $Mann - Whitney$ U-test was employed. Those results suggest that DeepGA-RNC performed similarly to PSO-RNC, VGG16, and VGG16-LSTM, despite having a significantly lower number of parameters (i.e., a competitive smaller model was obtained).

5 Discussion and Conclusion

The neuroevolution approach based on DeepGA has proven to be a good alternative, highlighting its potential for predicting steering angles. The study emphasizes the importance of exploring internal parameters and network structure to achieve smaller architectures with a competitive performance compared to state-of-the-art CNNs. The study has found that DeepGA-CNN, proposed as shown in Fig. 4, is an architecture with 35,972 parameters that can perform similarly to other complex models predicting steering angles. Getting smaller models favor the addition of possible explainable mechanisms and lead to include them in restricted hardware architectures. The integration of neuroevolution and the DeepGA algorithm [16] has shown promising results in predicting steering angles. Part of the future work considers using a multi-objective version of DeepGA to verify if better solutions can be obtained. Finally, experiments in a small car in controlled conditions is also expected. In future work, the intention is to work with established vision methods for angle prediction, as well as different compression methods to compare them with neuroevolution.

Acknowledgments. The first author acknowledges support of the National Council of Humanities, Science, and Technology (CONAHCYT) through a scholarship with no. CVU 1220680 to pursue graduate studies at Universidad Veracruzana.

References

1. Bojarski, M., et al.: End to end learning for self-driving cars (2016). https://doi.org/10.48550/arXiv.1604.07316
2. Bojarski, M., et al.: PilotNet: end-to-end learning for self-driving cars (2016). https://github.com/lhzlhz/PilotNet
3. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. CoRR (2017). http://arxiv.org/abs/1710.09282
4. Do, T.D., Duong, M.T., Dang, Q.V., Le, M.H.: Real-time self-driving car navigation using deep neural network, pp. 7–12 (2018). https://doi.org/10.1109/GTSD.2018.8595590
5. Galvãn, E., Mooney, P.: Neuroevolution in deep neural networks: current trends and future challenges. IEEE Trans. Artif. Intell. **2**(6), 476–493 (2021). https://doi.org/10.1109/TAI.2021.3067574
6. Gidado, U.M., Chiroma, H., Aljojo, N., Abubakar, S., Popoola, S.I., Al-Garadi, M.A.: A survey on deep learning for steering angle prediction in autonomous vehicles. IEEE Access **8**, 163797–163817 (2020). https://doi.org/10.1109/ACCESS.2020.3017883
7. Hwang, K., Park, J.H.: Steering control of an autonomous vehicle using CNN. J. Korean Inst. Inf. Commun. Eng. **24**, 834–841 (2020). https://api.semanticscholar.org/CorpusID:226447140
8. Khanum, A., Lee, C.Y., Yang, C.S.: End-to-end deep learning model for steering angle control of autonomous vehicles. In: 2020 International Symposium on Computer, Consumer and Control (IS3C), pp. 189–192 (2020). https://doi.org/10.1109/IS3C50286.2020.00056

9. Khidhir, Y.G., Morad, A.H.: Comparative transfer learning models for end-to-end self-driving car. Al-Khwarizmi Eng. J. **18**(4), 45–59 (2022). https://doi.org/10.22153/kej.2022.09.003, https://alkej.uobaghdad.edu.iq/index.php/alkej/article/view/814
10. Lorenzo, P.R., Nalepa, J.: Memetic evolution of deep neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 505–512. GECCO 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3205455.3205631
11. Papavasileiou, E., Cornelis, J., Jansen, B.: A systematic literature review of the successors of "neuroevolution of augmenting topologies". Evol. Comput. **29**(1), 1–73 (2021). https://doi.org/10.1162/evco_a_00282
12. Saleem, H., Riaz, F., Mostarda, L., Niazi, M.A., Rafiq, A., Saeed, S.: Steering angle prediction techniques for autonomous ground vehicles: a review. IEEE Access **9**, 78567–78585 (2021). https://doi.org/10.1109/ACCESS.2021.3083890
13. Saleem, H., et al.: Optimizing steering angle predictive convolutional neural network for autonomous car. Computers. Mater. Continua **71**(2), 2285–2302 (2022). https://doi.org/10.32604/cmc.2022.022726, http://www.techscience.com/cmc/v71n2/45840
14. Stanley, K.O., Clune, J., Lehman, J., Miikkulainen, R.: Designing neural networks through neuroevolution. Nat. Mach. Intell. **1**(1), 24–35 (2019). https://doi.org/10.1038/s42256-018-0006-z
15. Udacity: Udacity self-driving car datasets (Año). https://github.com/udacity/self-driving-car/tree/master/datasets
16. Vargas-Hákim, G.A., Mezura-Montes, E., Acosta-Mesa, H.G.: Hybrid encodings for neuroevolution of convolutional neural networks: a case study. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1762–1770. GECCO 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3449726.3463133
17. Wu, T., Luo, A., Huang, R., Cheng, H., Zhao, Y.: End-to-end driving model for steering control of autonomous vehicles with future spatiotemporal features. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 950–955 (2019). https://doi.org/10.1109/IROS40897.2019.8968453

Referencias

[1] H. Saleem, F. Riaz, L. Mostarda, M. A. Niazi, A. Rafiq y S. Saeed, «Steering Angle Prediction Techniques for Autonomous Ground Vehicles: A Review,» *IEEE Access*, vol. 9, págs. 78 567-78 585, 2021. DOI: 10.1109/ACCESS.2021.3083890.

[2] U. M. Gidado, H. Chiroma, N. Aljojo, S. Abubakar, S. I. Popoola y M. A. Al-Garadi, «A Survey on Deep Learning for Steering Angle Prediction in Autonomous Vehicles,» *IEEE Access*, vol. 8, págs. 163 797-163 817, 2020. DOI: 10.1109/Access.2020.3017883.

[3] E. Galván y P. Mooney, «Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges,» *IEEE Transactions on Artificial Intelligence*, vol. 2, n.º 6, págs. 476-493, 2021. DOI: 10.1109/TAI.2021.3067574.

[4] K. O. Stanley, J. Clune, J. Lehman y R. Miikkulainen, «Designing neural networks through neuroevolution,» *Nature Machine Intelligence*, vol. 1, n.º 1, págs. 24-35, 2019, ISSN: 2522-5839. DOI: 10.1038/s42256-018-0006-z. dirección: https://doi.org/10.1038/s42256-018-0006-z.

[5] M. Bojarski, D. D. Testa, D. Dworakowski et al., *End to End Learning for Self-Driving Cars*, 2016. DOI: 10.48550/arXiv.1604.07316. arXiv: 1604.07316 [cs.CV].

[6] T.-D. Do, M.-T. Duong, Q.-V. Dang y M.-H. Le, «Real-Time Self-Driving Car Navigation Using Deep Neural Network,» en *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, 2018, págs. 7-12. DOI: 10.1109/GTSD.2018.8595590.

[7] T. Wu, A. Luo, R. Huang, H. Cheng e Y. Zhao, «End-to-End Driving Model for Steering Control of Autonomous Vehicles with Future Spatiotemporal Features,» en *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, págs. 950-955. DOI: 10.1109/IROS40897.2019.8968453.

[8] A. Khanum, C.-Y. Lee y C.-S. Yang, «End-to-End Deep Learning Model for Steering Angle Control of Autonomous Vehicles,» en *2020 International Symposium on Computer, Consumer and Control (IS3C)*, 2020, págs. 189-192. DOI: 10.1109/IS3C50286.2020.00056.

[9] H. Jiang, L. Chang, Q. Li y D. Chen, «Deep Transfer Learning Enable End-to-End Steering Angles Prediction for Self-driving Car,» en *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, págs. 405-412. DOI: 10.1109/IV47402.2020.9304611.

[10] K. Hwang y ParkJin-Hyun, «Steering Control of an Autonomous Vehicle Using CNN,» *The Journal of the Korean Institute of Information and Communication Engineering*, vol. 24, págs. 834-841, 2020. dirección: https://api.semanticscholar.org/CorpusID:226447140.

[11] D. V. P. Mygapula, A. S, S. V. V. V y S. K. P, «CNN based End to End Learning Steering Angle Prediction for Autonomous Electric Vehicle,» en *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2021, págs. 1-6. DOI: 10.1109/ICECCT52121.2021.9616875.

[12] M. Z. Asy'ari, M. Phang, N. Suganda e Y. Mariana, «Intelligent Small Scale Autonomous Vehicle Development Based on Convolutional Neural Network (CNN) for Steering Angle Prediction,» en *Innovative Technologies in Intelligent Systems and Industrial Applications*, S. C. Mukhopadhyay, S. N. A. Senanayake y P. C. Withana, eds., Cham: Springer Nature Switzerland, 2023, págs. 3-12. DOI: 10.1007/978-3-031-29078-7_1.

[13] L. Chi e Y. Mu, *Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues*, 2017. arXiv: 1708.03798 [cs.CV].

[14] Udacity. «Udacity Self-Driving Car Datasets.» (Año), dirección: https://github.com/udacity/self-driving-car/tree/master/datasets.

[15] P. J. Navarro, L. Miller, F. Rosique, C. Fernández-Isla y A. Gila-Navarro, «End-to-End Deep Neural Network Architectures for Speed and Steering Wheel Angle Prediction in Autonomous Driving,» *Electronics*, vol. 10, n.º 11, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10111266. dirección: https://www.mdpi.com/2079-9292/10/11/1266.

[16] M. G. Bechtel, E. Mcellhiney, M. Kim y H. Yun, «DeepPicar: A Low-Cost Deep Neural Network-Based Autonomous Car,» en *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, págs. 11-21. DOI: 10.1109/RTCSA.2018.00011.

[17] Y. G. Khidhir y A. H. Morad, «Comparative Transfer Learning Models for End-to-End Self-Driving Car,» *Al-Khwarizmi Engineering Journal*, vol. 18, n.º 4, 45–59, 2022. DOI: 10.22153/kej.2022.09.003. dirección: https://alkej.uobaghdad.edu.iq/index.php/alkej/article/view/814.

[18] H. Saleem, F. Riaz, A. Shaikh et al., «Optimizing Steering Angle Predictive Convolutional Neural Network for Autonomous Car,» *Computers, Materials & Continua*, vol. 71, n.º 2, págs. 2285-2302, 2022, ISSN: 1546-2226. DOI: 10.32604/cmc.2022.022726. dirección: http://www.techscience.com/cmc/v71n2/45840.

[19] M. K. Islam, M. N. Yeasmin, C. Kaushal, M. A. Amin, M. R. Islam y M. I. Hossain Showrov, «Comparative Analysis of Steering Angle Prediction for Automated Object using Deep Neural Network,» en *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2021, págs. 1-7. DOI: 10.1109/ICRITO51393.2021.9596499.

[20] A. AbuZekry, I. Sobh, M. Hadhoud y M. Fayek, «Comparative study of NeuroEvolution algorithms in reinforcement learning for self-driving cars,» *European Journal of Engineering Science and Technology*, vol. 2, n.º 4, págs. 60-71, 2019. DOI: 10.33422/EJEST.2019.09.38.

[21] S. M. J. Jalali, P. M. Kebria, A. Khosravi, K. Saleh, D. Nahavandi y S. Nahavandi, «Optimal Autonomous Driving Through Deep Imitation Learning and Neuroevolution,» en *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, págs. 1215-1220. DOI: 10.1109/SMC.2019.8914582.

[22] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo y A. Ylä-Jääski, «Latency and throughput characterization of convolutional neural networks for mobile computer vision,» en *Proceedings of the 9th ACM Multimedia Systems Conference*, ép. MMSys '18, Amsterdam, Netherlands: Association for Computing Machinery, 2018, 204–215, ISBN: 9781450351928. DOI: 10.1145/3204949.3204975. dirección: https://doi.org/10.1145/3204949.3204975.

[23] M. Ji, Z. Al-Ars, P. Hofstee, Y. Chang y B. Zhang, «FPQNet: Fully Pipelined and Quantized CNN for Ultra-Low Latency Image Classification on FPGAs Using OpenCAPI,» *Electronics*, vol. 12, n.º 19, 2023. DOI: 10.3390/electronics12194085.

[24] B. Yan, Q. Liu, J. Shen, D. Liang, B. Zhao y L. Ouyang, «A survey of low-latency transmission strategies in software defined networking,» *Computer Science Review*, vol. 40, pág. 100 386, 2021, ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2021.100386. dirección: https://www.sciencedirect.com/science/article/pii/S1574013721000265.

[25] G. Ascia, V. Catania, J. Jose, S. Monteleone, M. Palesi y D. Patti, «Improving Inference Latency and Energy of Network-on-Chip based Convolutional Neural Networks through Weights Compression,» en *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020, págs. 54-63. DOI: 10.1109/IPDPSW50202.2020.00017.

[26] L. Liao, H. Li, W. Shang y L. Ma, «An Empirical Study of the Impact of Hyperparameter Tuning and Model Optimization on the Performance Properties of Deep Neural Networks,» *ACM Trans. Softw. Eng. Methodol.*, vol. 31, n.º 3, 2022, ISSN: 1049-

331X. DOI: 10.1145/3506695. dirección: https://doi.org/10.1145/3506695.

[27] P. Dasun Dileepa Pitawela y L. Ranathunga, «Low Latency Approach in Road Sign Recognition and Tracking for Autonomous Vehicles,» en *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2018, págs. 1015-1022. DOI: 10.1109/FSKD.2018.8687151.

[28] A. Mujtaba, W.-K. Lee y S. O. Hwang, «Low Latency Implementations of CNN for Resource-Constrained IoT Devices,» *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, n.º 12, págs. 5124-5128, 2022. DOI: 10.1109/TCSII.2022.3205029.

[29] On-Road Automated Driving (ORAD) Committee, «Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,» SAE International, inf. téc., 2018, Issuing Committee: On-Road Automated Driving (ORAD) Committee, pág. 35. DOI: 10.4271/J3016_201806. dirección: https://doi.org/10.4271/J3016_201806.

[30] Society of Automotive Engineers, «Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,» *SAE Standard J3016*, 2021. DOI: 10.4271/J3016_202104.

[31] C. C. et al., «LiDAR Point Cloud Processing for Autonomous Driving: A Survey,» *IEEE Transactions on Intelligent Vehicles*, vol. 4, n.º 4, págs. 513-539, 2019. DOI: 10.1109/TIV.2019.2953414.

[32] J. W. et al., «Real-time LiDAR Data Processing for Autonomous Driving,» *IEEE Transactions on Intelligent Vehicles*, vol. 5, n.º 1, págs. 5-20, 2020. DOI: 10.1109/TIV.2019.2961355.

[33] S. S. et al., «Autonomous Vehicle Control in Urban Environments: Challenges and Solutions,» *IEEE Transactions on Intelligent Vehicles*, vol. 4, n.º 1, págs. 2-15, 2019. DOI: 10.1109/TIV.2019.2895836.

[34] M. Harrer y P. Pfeffer, eds., *Steering Handbook*, 1.ª ed. Springer Cham, 2016, págs. XVII, 565, https://doi.org/10.1007/978-3-319-05449-0. DOI: 10.1007/978-3-319-05449-0.

[35] WapCar. «Variable Steering Ratio.» Imagen tomada de WapCar. (Año de la imagen), dirección: https://www.wapcar.my/carpedia/do-you-know-the-variable-steering-ratio-299.

[36] A. G. et al., «Automatic and Robust Lane Detection for Autonomous Vehicles,» *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, n.º 2, págs. 472-482, 2015. DOI: 10.1109/TITS.2014.2349939.

[37] D. N. et al., «Robust Road Lane Detection and Tracking with RANSAC and Kalman Filter,» *IEEE Transactions on Intelligent Vehicles*, vol. 5, n.º 1, págs. 47-59, 2020. DOI: 10.1109/TIV.2019.2935186.

[38] J.-S. Zhao, X Liu, Z.-J. Feng y J. Dai, «Design of an Ackermann-type steering mechanism,» *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 227, n.º 11, págs. 2549-2562, 2013. DOI: 10.1177/0954406213475980.

[39] X. L. et al., «Deep Learning-Based Lane Detection for Autonomous Driving: A Comprehensive Review,» *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, n.º 11, págs. 5065-5079, 2021. DOI: 10.1109/TNNLS.2020.3027241.

[40] L. Y. et al., «Improving the Robustness of Lane Detection Models for Autonomous Vehicles,» *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, n.º 5, págs. 2821-2830, 2021. DOI: 10.1109/TITS.2020.3027890.

[41] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman y N. Muhammad, «A Survey of End-to-End Driving: Architectures and Training Methods,» *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, n.º 4, págs. 1364-1384, 2022. DOI: 10.1109/TNNLS.2020.3043505.

[42] M. H. Z. et al., «Integrating Autonomous Vehicles into Urban Traffic Networks: A Review,» *IEEE Transactions on Intelligent Vehicles*, vol. 6, n.º 1, págs. 101-113, 2021. DOI: 10.1109/TIV.2021.3051378.

[43] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook, A Textbook*, 1.ª ed. Springer Cham, 2018, págs. XXIII, 497, Springer International Publishing AG, part of Springer Nature. DOI: 10.1007/978-3-319-94463-0. dirección: https://doi.org/10.1007/978-3-319-94463-0.

[44] E. H. Jeon, «Multiple regression,» en *Advancing quantitative methods in second language research*, Routledge, 2015, págs. 131-158.

[45] P. Fergus y C. Chalmers, «Performance Evaluation Metrics,» en *Applied Deep Learning: Tools, Techniques, and Implementation*. Cham: Springer International Publishing, 2022, págs. 115-138, ISBN: 978-3-031-04420-5. DOI: 10.1007/978-3-031-04420-5_5. dirección: https://doi.org/10.1007/978-3-031-04420-5_5.

[46] «MAPE (mean absolute percentage error)MEAN ABSOLUTE PERCENTAGE ERROR (MAPE),» en *Encyclopedia of Production and Manufacturing Management*, P. M. Swamidass, ed. Boston, MA: Springer US, 2000, págs. 462-462, ISBN: 978-1-4020-0612-8. DOI: 10.1007/1-4020-0612-8_580. dirección: https://doi.org/10.1007/1-4020-0612-8_580.

[47] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, «Gradient-based learning applied to document recognition,» *Proceedings of the IEEE*, vol. 86, n.º 11, págs. 2278-2324, 1998. DOI: 10.1109/5.726791.

[48] L. Alzubaidi, J. Zhang, A. J. Humaidi et al., «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,» *Journal of Big Data*, vol. 8, n.º 1, pág. 53, 2021. DOI: 10.1186/s40537-021-00444-8. dirección: https://doi.org/10.1186/s40537-021-00444-8.

[49] A. Krizhevsky, I. Sutskever y G. E. Hinton, «Imagenet classification with deep convolutional neural networks,» en *Advances in neural information processing systems*, 2012, págs. 1097-1105.

[50] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 4th. Pearson Education Limited, 2018, ISBN: 978-1-292-22304-9.

[51] S. Ioffe y C. Szegedy, «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,» en *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach y D. Blei, eds., ép. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, págs. 448-456. dirección: https://proceedings.mlr.press/v37/ioffe15.html.

[52] D. Scherer, A. Müller y S. Behnke, «Evaluation of pooling operations in convolutional architectures for object recognition,» en *International conference on artificial neural networks*, Springer, 2010, págs. 92-101. DOI: 10.1007/978-3-642-15825-4_10.

[53] I. Goodfellow, Y. Bengio y A. Courville, *Deep learning*. MIT press, 2016.

[54] Y. Bengio, A. Courville y P. Vincent, «Representation learning: A review and new perspectives,» *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, n.º 8, págs. 1798-1828, 2013. DOI: 10.1109/TPAMI.2013.50.

[55] X. Glorot e Y. Bengio, «Understanding the difficulty of training deep feedforward neural networks,» en *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh y M. Titterington, eds., ép. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, págs. 249-256. dirección: https://proceedings.mlr.press/v9/glorot10a.html.

[56] V. Nair y G. E. Hinton, «Rectified linear units improve restricted boltzmann machines,» en *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, págs. 807-814.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov, «Dropout: A simple way to prevent neural networks from overfitting,» *The journal of machine learning research*, vol. 15, n.º 1, págs. 1929-1958, 2014.

[58] S. Ioffe y C. Szegedy, «Batch normalization: Accelerating deep network training by reducing internal covariate shift,» en *International conference on machine learning*, PMLR, 2015, págs. 448-456. DOI: 10.48550/arXiv:1502.03167.

[59] C. Chen, A. Seff, A. Kornhauser y J. Xiao, «DeepDriving: Learning affordance for direct perception in autonomous driving,» en *Proceedings of the IEEE International Conference on Computer Vision*, 2015, págs. 2722-2730. DOI: 10.1109/ICCV.2015.312.

[60] E. Li, L. Zeng, Z. Zhou y X. Chen, «Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing,» *IEEE Transactions on Wireless Communications*, vol. 19, n.º 1, págs. 447-457, 2020. DOI: 10.1109/TWC.2019.2946140.

[61] S. Hauschild y H. Hellbrück, «Latency and Energy Consumption of Convolutional Neural Network Models from IoT Edge Perspective,» en *Internet of Things*, A. González-Vidal, A. Mohamed Abdelgawad, E. Sabir, S. Ziegler y L. Ladid, eds., Cham: Springer International Publishing, 2022, págs. 385-396, ISBN: 978-3-031-20936-9. DOI: 10.1007/978-3-031-20936-9_31.

[62] M. M. Yesuf y B. G. Assefa, «Model Compression Techniques in Deep Neural Networks,» en *Pan-African Conference on Artificial Intelligence*, T. Girma Debelee, A. Ibenthal y F. Schwenker, eds., Cham: Springer Nature Switzerland, 2023, págs. 169-190, ISBN: 978-3-031-31327-1. DOI: 10.1007/978-3-031-31327-1_10.

[63] M. Wistuba, A. Rawat y T. Pedapati, «A Survey on Neural Architecture Search,» *CoRR*, vol. abs/1905.01392, 2019. arXiv: 1905.01392. dirección: http://arxiv.org/abs/1905.01392.

[64] B. Zoph y Q. V. Le, «Neural Architecture Search with Reinforcement Learning,» *CoRR*, vol. abs/1611.01578, 2016. arXiv: 1611.01578. dirección: http://arxiv.org/abs/1611.01578.

[65] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen y K. C. Tan, «A Survey on Evolutionary Neural Architecture Search,» *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, n.º 2, págs. 550-570, 2023. DOI: 10.1109/TNNLS.2021.3100554.

[66] S. Santra, J.-W. Hsieh y C.-F. Lin, «Gradient Descent Effects on Differential Neural Architecture Search: A Survey,» *IEEE Access*, vol. 9, págs. 89 602-89 618, 2021. DOI: 10.1109/ACCESS.2021.3090918.

[67] X. Zhou, A. K. Qin, Y. Sun y K. C. Tan, «A Survey of Advances in Evolutionary Neural Architecture Search,» en *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, págs. 950-957. DOI: 10.1109/CEC45853.2021.9504890.

[68] G. Karafotias, M. Hoogendoorn y A. E. Eiben, «Parameter Control in Evolutionary Algorithms: Trends and Challenges,» *IEEE Transactions on Evolutionary Computation*, vol. 19, n.º 2, págs. 167-187, 2015. DOI: 10.1109/TEVC.2014.2308294.

[69] D. Floreano, P. Dürr y C. Mattiussi, «Neuroevolution: from architectures to learning,» *Evolutionary Intelligence*, vol. 1, n.º 1, págs. 47-62, 2008, ISSN: 1864-5917. DOI: 10.1007/s12065-007-0002-4. dirección: https://doi.org/10.1007/s12065-007-0002-4.

[70] K. O. Stanley, J. Clune, J. Lehman y R. Miikkulainen, «Designing neural networks through neuroevolution,» *Nature Machine Intelligence*, vol. 1, n.º 1, págs. 24-35, 2019. DOI: 10.1038/s42256-018-0006-z.

[71] G.-A. Vargas-Hákim, E. Mezura-Montes y H.-G. Acosta-Mesa, «Hybrid Encodings for Neuroevolution of Convolutional Neural Networks: A Case Study,» en *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ép. GECCO '21, Lille, France: Association for Computing Machinery, 2021, 1762–1770, ISBN: 9781450383516. DOI: 10.1145/3449726.3463133. dirección: https://doi.org/10.1145/3449726.3463133.

[72] B. Wang, Y. Sun, B. Xue y M. Zhang, «A Hybrid GA-PSO Method for Evolving Architecture and Short Connections of Deep Convolutional Neural Networks,» en *PRICAI 2019: Trends in Artificial Intelligence*, A. C. Nayak y A. Sharma, eds., Cham: Springer International Publishing, 2019, págs. 650-663, ISBN: 978-3-030-29894-4. DOI: 10.1007/978-3-030-29894-4_52.

[73] M. Bojarski, D. D. Testa, D. Dworakowski et al. «PilotNet: End-to-End Learning for Self-Driving Cars.» (2016), dirección: https://github.com/lhzlhz/PilotNet.

[74] Y. Gong, *ImageAngle-Udacity (IA-Udacity)*, 2023. DOI: 10.21227/1evd-ew07. dirección: https://dx.doi.org/10.21227/1evd-ew07.

[75] L. W. Turner, «Practical Tourism Forecasting,» *International Journal of Forecasting*, vol. 13, n.º 2, págs. 296-297, 1997, "Practical tourism forecasting: D.C. Frechtling, 1996, (Butterworth Heineman, Oxford), + 240 pp., Softcover, ISBN 0 7506 0877 3".

[76] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[77] K. P. Seng, P. J. Lee y L. M. Ang, «Embedded Intelligence on FPGA: Survey, Applications and Challenges,» *Electronics*, vol. 10, n.º 8, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10080895. dirección: https://www.mdpi.com/2079-9292/10/8/895.

[78] H. V. Pham, T. G. Tran, C. D. Le, A. D. Le y H. B. Vo, «Benchmarking Jetson Edge Devices with an End-to-End Video-Based Anomaly Detection System,» en *Advances in Information and Communication*. Springer Nature Switzerland, 2024, 358–374, ISBN: 9783031539633. DOI: 10.1007/978-3-031-53963-3_25. dirección: http://dx.doi.org/10.1007/978-3-031-53963-3_25.

[79] A. Yazdanbakhsh, K. Seshadri, B. Akin, J. Laudon y R. Narayanaswami, «An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks,» *CoRR*, vol. abs/2102.10423, 2021. arXiv: 2102.10423. dirección: https://arxiv.org/abs/2102.10423.

[80] F. Daghero, D. Jahier Pagliari y M. Poncino, «Energy-efficient deep learning inference on edge devices,» en Academic Press, sep. de 2020, ISBN: 9780128231234. DOI: 10.1016/bs.adcom.2020.07.002.

[81] G. Abraham y M. Nithya, «Multi-Functional Personal Assistant Robot Using Raspberry Pi and Coral Accelerator,» en *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021, págs. 638-643. DOI: 10.1109/ICCMC51019.2021.9418299.

[82] W. Niu, M. M. R. Sanim, Z. Shu et al., «SmartMem: Layout Transformation Elimination and Adaptation for Efficient DNN Execution on Mobile,» en *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ép. ASPLOS '24, La Jolla, CA, USA: Association for Computing Machinery, 2024, 916–931, ISBN: 9798400703867. DOI: 10.1145/3620666.3651384. dirección: https://doi.org/10.1145/3620666.3651384.

[83] X. Yang, C. Zhuang, W. Feng, Z. Yang y Q. Wang, «FPGA Implementation of a Deep Learning Acceleration Core Architecture for Image Target Detection,» *Applied Sciences*, vol. 13, n.º 7, 2023, ISSN: 2076-3417. DOI: 10.3390/app13074144. dirección: https://www.mdpi.com/2076-3417/13/7/4144.

[84] J.-D. Velazco-Muñoz, H.-G. Acosta-Mesa y E. Mezura-Montes, «Reducing Parameters by Neuroevolution in CNN for Steering Angle Estimation,» en *Pattern Recognition*, 2024, págs. 377-386. DOI: 10.1007/978-3-031-62836-8_35.