UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

Maestría en

MÉTODOS MATEMÁTICOS Y SIMULACIÓN NUMÉRICA EN INGENIERÍA

RPC-SO-42-NO.769-2019

Opción de Titulación:

Artículos profesionales de alto nivel

Tema:

Análisis de algoritmos optimizadores de función de coste en el rendimiento de una red neuronal convolucional YOLO aplicada a la detección automática de peatones

Autor(es)

Bryan Gustavo Paredes Ballesteros

Director:

Paulina Adriana Morillo Alcívar

QUITO – Ecuador

2023

Autor(es):

Bryan Gustavo Paredes Ballesteros
Ingeniero Mantenimiento Automotriz
Candidato a Magíster en Métodos Matemáticos y Simulación Numérica en Ingeniería por la Universidad Politécnica Salesiana – Sede Quito.
bryangustavoparedes@gmail.com

Dirigido por:

Paulina Adriana Morillo Alcívar
Ingeniera en Electrónica y Telecomunicaciones
Magister en Gestión de la Información
pmorillo@ups.edu.ec

BRYAN GUSTAVO PAREDES BALLESTEROS
ANÁLISIS DE ALGORITMOS OPTIMIZADORES DE FUNCIÓN DE COSTE EN EL RENDIMIENTO DE UNA RED NEURONAL CONVOLUCIONAL YOLO APLICADA A LA DETECCIÓN AUTOMÁTICA DE PEATONES.

ANÁLISIS DE ALGORITMOS OPTIMIZADORES DE FUNCIÓN DE COSTE EN EL RENDIMIENTO DE UNA RED NEURONAL CONVOLUCIONAL YOLO APLICADA A LA DETECCIÓN AUTOMÁTICA DE PEATONES

Autor(es):

BRYAN GUSTAVO PAREDES BALLESTEROS

Resumen

En los últimos años, las investigaciones en el campo del Deep Learning han adquirido una gran relevancia en diversas aplicaciones de la vida real, incluyendo la conducción autónoma y la detección de objetos asociada a esta área. Para llevar a cabo este proceso, es necesario entrenar redes neuronales convolucionales capaces de realizar detecciones rápidas y precisas. En este trabajo se muestra el análisis comparativo del rendimiento de la red YOLOv5, entrenada con tres diferentes optimizadores de función de costo Adam, SGD y AdamW. Los algoritmos optimizadores de función de costo desempeñan un papel fundamental en el entrenamiento y el rendimiento de las redes neuronales, ya que la función de costo cuantifica la discrepancia entre los valores reales y los valores predichos por el modelo. Los experimentos se llevaron a cabo con el set de datos libre Self Driving Car Dataset de Udacity y la versión X-Large de YOLO. Para el análisis comparativo se realizó la prueba de comparaciones múltiples de Kruskal-Wallis y la prueba post hoc de Dunn de las que se desprende la conclusión que el optimizador SGD difiere en gran medida a Adam y AdamW. A pesar de que los resultados de un modelo de DL no son generalizables, los resultados muestran que SGD consiguió métricas de rendimiento significativamente mayores que Adam y AdamW. Logrando una precisión promedio mayor al 90%. Estos resultados destacan la importancia de seleccionar cuidadosamente el optimizador de función de costo en el entrenamiento de redes neuronales convolucionales, como YOLOv5, con el fin de mejorar el rendimiento y la precisión en tareas de detección de objetos, como en el contexto de la conducción autónoma.

Palabras clave:

Red Neuronal YOLOv5; optimizadores de función de coste; Deep learning; Análisis comparativo

Abstract

In recent years, research in the discipline of Deep Learning has improved great relevance in various real-life applications, including autonomous driving and object detection associated with this area. To carry out this process, it is necessary to train convolutional neural networks capable of performing fast and accurate detections. This work presents a comparative analysis of the performance of the YOLOv5 network, trained with three different cost function optimizers: Adam, SGD, and AdamW. The cost function optimization algorithms play a fundamental role in the training and performance of neural networks, as the cost function quantifies the discrepancy between actual values and values predicted by the model. The experiments were run using the Self Driving Car Dataset from Udacity and the X-Large version of YOLO. For the comparative analysis, the Kruskal-Wallis multiple comparisons test and the Dunn post hoc test were performed, from which the conclusion is drawn that the SGD optimizer differs significantly from Adam and AdamW. Although the results of a DL model are not generalizable, the results show that SGD achieved higher performance metrics than Adam and AdamW, achieving an average precision greater than 90%. These results highlight the importance of carefully selecting the cost function optimizer in the training of convolutional neural networks, such as YOLOv5, to improve performance and accuracy in object detection tasks, such as in the context of autonomous driving.

Keywords:

Neural Network YOLOv5, loss function optimizers, Deep Learning, Comparative Analysis

1. Introducción

En los últimos años, el campo de la inteligencia artificial (IA) ha experimentado avances significativamente notables. Gracias a técnicas de Aprendizaje Profundo (*Deep Learning*) las computadoras pueden memorizar y aprender características de diversos tipos de datos, como video, texto, sonido, entre otros (S. Zhang et al., 2021). En este sentido, las redes neuronales profundas se han convertido en herramientas poderosas para aprender de forma automática patrones y características complejas presentes en los conjuntos de datos (Brownlee, 2019). Por ejemplo, las redes neuronales recurrentes (RNN) se usan en el campo del procesamiento natural del lenguaje para el reconocimiento de voz, la traducción automática, la generación de texto, el análisis de sentimientos en texto, la generación de música, etc. Así mismo, las redes neuronales convolucionales (CNN) se usan en el campo de la visión por computador para la clasificación de imágenes, detección de objetos, segmentación semántica, reconocimiento facial, detección de anomalías, entre otras (Xin et al., 2020).

En el campo de la conducción autónoma las redes neuronales convolucionales juegan un papel fundamental para que los vehículos autónomos naveguen de manera segura y eficiente, ya que permiten mejorar la percepción del entorno, la toma de decisiones y la seguridad en general. Las tareas más comunes que se realizan son la detección y clasificación de objetos en tiempo real como vehículos, peatones y señales de tráfico (Zhang et al., 2021). De acuerdo a (Meena et al., 2022), existen varios tipos de arquitecturas de redes neuronales convolucionales (CNN) como pueden ser: RetinaNet, Faster R-CNN, Single-Shot Detector y You Only Look Once (YOLO). En este contexto YOLO puede detectar una amplia variedad de objetos, como peatones, vehículos, señales de tráfico y bicicletas, etc. Su principal ventaja frente a otras arquitecturas es su capacidad para realizar detecciones rápidas y en tiempo real (Verma et al., 2021), lo que es crucial para garantizar una

respuesta rápida y precisa del vehículo autónomo. Al proporcionar información en tiempo real sobre los objetos presentes en el entorno, el sistema de conducción autónoma puede planificar y ajustar su ruta y velocidad de manera segura.

En términos generales, una red neuronal convolucional (CNN) es un modelo que consta de múltiples capas, incluyendo capas convolucionales, de agrupación y completamente conectadas. Durante el entrenamiento, el objetivo es ajustar los pesos y sesgos de la red para realizar predicciones precisas en nuevos datos de entrada (Campesato, 2020; Pérez González, 2021). Estos pesos se optimizan minimizando una función de pérdida, que cuantifica la discrepancia entre las predicciones de la red y los valores reales del conjunto de datos de entrenamiento. Esta función de pérdida no solo determina la capacidad de la red para realizar predicciones precisas durante la fase de entrenamiento, sino también su rendimiento posterior en la fase de producción.

A pesar del progreso significativo en el campo de la visión por computadora, que ha permitido el desarrollo de sistemas de conducción autónoma altamente precisos (Kaszas & Roberts, 2022; Lakshmanan et al., 2021), el rendimiento de los modelos de detección de objetos en tiempo real sigue siendo una preocupación para los fabricantes de vehículos, quienes deben cumplir con estándares de seguridad vial exigentes. En este sentido, la detección de peatones es especialmente crucial para prevenir accidentes y salvaguardar vidas humanas, además de cumplir con los objetivos de los sistemas de asistencia avanzada al conductor, ADAS, por sus siglas en inglés *Advanced Driver Assistance Systems* (Perumal et al., 2021).

Este artículo presenta una comparación del rendimiento de la CNN YOLO en la detección automática de peatones, considerando diferentes algoritmos de optimización. Los algoritmos de optimización son elementos clave dentro de un conjunto de parámetros, que incluyen el tamaño de entrada, umbral de confianza, tasa de aprendizaje, número de capas, entre otros ajustes configurables, y en conjunto, determinan el desempeño del modelo. Los algoritmos de optimización comúnmente utilizadas en CNNs son: Descenso de gradiente estocástico (SGD),

Adam (*Adaptive Moment Estimation*), RMSprop (Root Mean Square Propagation) y AdaGrad (*Adaptive Gradient*). Estos algoritmos desempeñan un papel fundamental en el entrenamiento de la red y contribuyen a la búsqueda de pesos óptimos que maximicen la precisión en la detección de peatones (Ge & Dou, 2023; Ranjan & Senthamilarasu, 2020).

2. Trabajos Relacionados

En el ámbito de las redes neuronales convolucionales (CNN), los optimizadores desempeñan un papel crucial para lograr un rendimiento óptimo en tareas como la detección de objetos en tiempo real. En este contexto, la investigación de (Pomerat et al., 2019), realiza una comparativa entre diferentes funciones de activación y optimizadores de función de pérdida en una red neuronal, y los compara con los resultados obtenidos mediante regresión polinomial en *Machine Learning*. El artículo destaca la popularidad y el rendimiento del optimizador SGD (*Stochastic Gradient Descent*) en comparación con los optimizadores Adam y RMSProp. Además, los investigadores llevaron a cabo experimentos para encontrar la mejor configuración entre un optimizador y una función de activación. Concluyendo que el optimizador Adam presenta un alto rendimiento cuando se utiliza la función de activación Sigmoide en la red. Por otro lado, el optimizador SGD muestra una gran compatibilidad con la función de activación ReLU.

En otros trabajos como el de (Vani & Rao, 2019) se proporciona un análisis detallado de la aplicación de los optimizadores y evalúa su rendimiento utilizando diversas métricas. Por ejemplo, en cuanto a la precisión, los autores concluyeron que los optimizadores Adamax, Nadam y RMSProp demostraron tener una ligera ventaja sobre los demás. Asimismo, se evaluó la métrica de *recall*, donde nuevamente los mismos optimizadores mencionados anteriormente se posicionaron en los primeros lugares. En consecuencia, los resultados de esta investigación llevaron a la conclusión definitiva de que el mejor algoritmo optimizador de función de costo es Adamax, con una precisión alcanzando el 99.5%.

De manera similar, la investigación de (Pathak et al., 2020) propone un análisis comparativo de optimizadores de función de coste, en una red neuronal profunda aplicada a la clasificación de naranjas, los optimizadores propuestos fueron: Descenso de Gradiente, Descenso de Gradiente Estocástico (SGD) con *momentum*,

RMSProp y Adam. Los autores realizaron un experimento que consistió en clasificar diferentes tipos de naranjas, con el uso de varias funciones de activación, optimizadores y funciones de costo con entropía cruzada. De este modo, determinaron la precisión del modelo y concluyeron que el optimizador con mayor rendimiento fue SGD con *momentum* alcanzando un 95%.

En el área de la microbiología también se han realizado estudios que incluyen el uso de CNNs. Así, el trabajo de (Shabrina et al., 2023) propone el diseño de varios modelos de *Deep Learning* para la identificación de parásitos que afectan a las plantas. Los investigadores entrenaron varias CNNs con tres diferentes optimizadores (Adam, SGD y RMSProp). Además, se utilizaron diferentes condiciones de entorno en la preparación de las imágenes de entrenamiento. De acuerdo con los resultados, el modelo desarrollado denominado EfficientNtV2M con el uso del optimizador RMSProp obtuvo la mayor precisión con un 97.94%, F1-score del 98.26% y un mAP (*mean average precision*) del 97.99%.

En cuanto a YOLO, su uso se ha extendido ampliamente a diversas aplicaciones, incluyendo el campo militar, donde se utiliza para detectar objetivos militares mediante el análisis de imágenes capturadas por UAVs o drones. En un estudio reciente realizado por (Kumar & Kumar, 2023), se propuso entrenar la red YOLO utilizando dos optimizadores de función de pérdida: SGD y Adam. Posteriormente, se ajustaron los hiperparámetros de ambos optimizadores para determinar el modelo óptimo en la detección de objetos. Los resultados obtenidos revelaron que el optimizador SGD, con un *momentum* de 0.801, mostró un rendimiento superior en la detección de objetos a alturas específicas. Este estudio demuestra la importancia de la selección adecuada de los optimizadores de función de pérdida en el entrenamiento de modelos YOLO, y proporciona una base sólida para mejorar la precisión y el rendimiento de detección en diferentes contextos de aplicación.

3. Materiales y Métodos

3.1 Materiales

Para el desarrollo de la investigación, se utilizó el *set* de imágenes para experimentos de conducción autónoma provisto por Udacity, denominado *Open Source Self-Driving Car*. Es un conjunto que comprende 97942 etiquetas en 11 clases, y 15000 imágenes para entrenamiento, validación y pruebas (Udacity, 2018). El conjunto de imágenes contiene 10806 etiquetas de peatones, de este, se utilizó 2319 imágenes para entrenamiento y 681 imágenes para validación. Adicionalmente, se empleó la plataforma RoboFlow para el procesamiento de las etiquetas de las imágenes.

Respecto a los equipos usados para el procesamiento fue necesario acceder a procesamiento en *cloud* a través de Google Colaboratory, debido a que el entrenamiento y validación de una CNN consume altos recursos computacionales. El equipo con el que se entrenó las CNNs constó de una memoria RAM de 40 GB y un chip GPU NVIDIA A100-SXM4 con 6912 núcleos CUDA. Para el almacenamiento del set de datos se utilizó Google Drive. El peso del conjunto de imágenes fue de 630 MB.

El desarrollo de los experimentos se realizó en el entorno de Anaconda con la versión de Python 3.10.11. La Tabla 1 detalla las librerías empleadas en este trabajo.

Librería	Versión
Pandas	1.5.3
Numpy	1.24.3
Matplotlib	3.7.1
Seaborn	0.12.2
Scipy	1.10.1
Scikit-posthocs	0.7.0

Tabla 1 Librerías de Python utilizadas

3.2 Metodología

El desarrollo de este proyecto consta de tres fases que se detallan en el diagrama de la Figura 1.

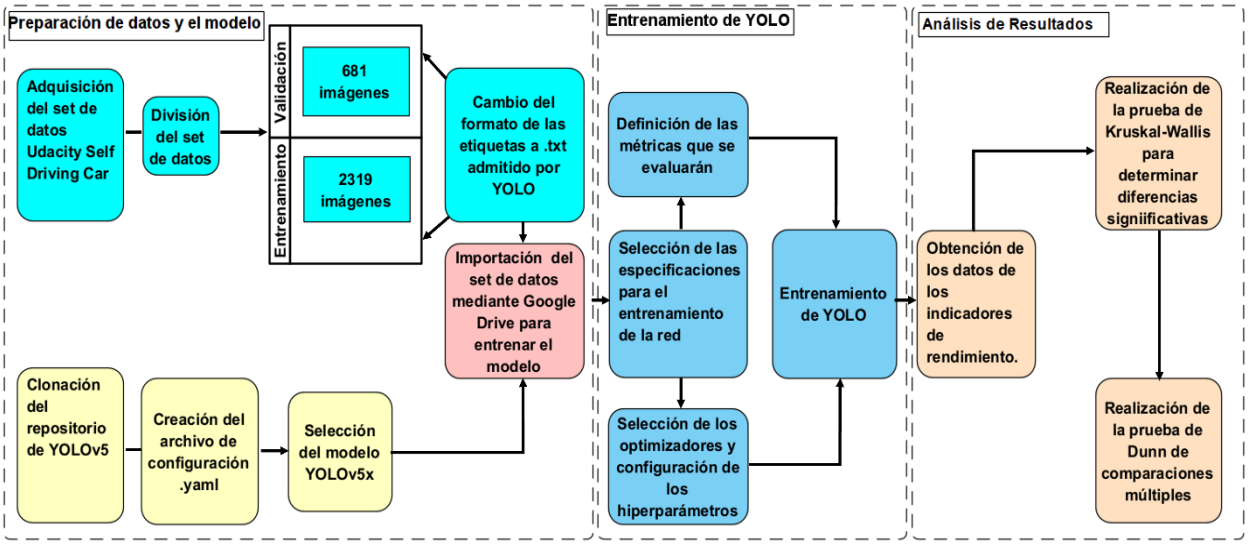

Figura 1 Metodología del entrenamiento de los modelos YOLO.

3.2.1 Preparación de las imágenes y el modelo de YOLO

Dado que el éxito o fracaso de un modelo de aprendizaje automático y de aprendizaje profundo viene dado por el conjunto de imágenes de entrenamiento, es necesario hacer una buena selección de las imágenes en función del objetivo que se pretende conseguir (A. Mathew et al., 2021). En este trabajo se usaron imágenes con peatones debidamente etiquetadas, de las cuales el 77% se usaron en el entrenamiento, y el 23% restante en la validación. Las imágenes de entrenamiento proporcionan las características para que el modelo pueda aprender y tenga la capacidad de generalización, de modo que pueda hacer predicciones sobre imágenes completamente nuevas (Huang et al., 2019). Es importante destacar que la investigación no realiza la evaluación con un *set* de prueba, esto debido a que el *set* de validación generaliza en gran medida las imágenes que el modelo no ha accedido previamente (Gu & Sun, 2023; Li et al., 2022; Sudars et al., 2022).

Una vez que se seleccionan las imágenes, se procede a cambiar el formato del archivo de las etiquetas. Las etiquetas de las imágenes originalmente se encuentran en formato JSON (Java Script Object Notation), pero, de acuerdo a (Rahman et al., 2019) la arquitectura de YOLO admite las etiquetas de tipo TXT, por esa razón, se utiliza la herramienta RoboFlow para la exportación en el formato adecuado de las etiquetas de las imágenes de los conjuntos de entrenamiento y validación. El correcto etiquetado de las imágenes juega un papel importante en el entrenamiento de la red, debido a que la etiqueta contiene información sobre la localización de las cajas delimitadoras y la clase a la que pertenece el objeto a detectar (M. P. Mathew & Mahesh, 2022).

Por otro lado, es necesario preparar el modelo YOLOv5 antes del entrenamiento, para ello se realiza una clonación del repositorio oficial de YOLOv5 en GitHub. De acuerdo a los desarrolladores de la red (Ultralytics, 2022), se debe modificar el archivo de configuración .yaml. En este archivo, se debe establecer la clase peatón como objetivo de la detección del modelo entrenado y también la organización del directorio dónde se localiza el set de datos que se importa de Google Drive.

Antes de efectuar el entrenamiento, se selecciona el modelo YOLOv5x que es parte de la familia de la versión quinta de YOLO (Ultralytics, 2022). YOLOv5x es más complejo y contiene un mayor número de conexiones entre nodos lo que mejora su capacidad de aprendizaje. Además, ocupa una memoria de 166 MB.

Debido a que se utiliza computo en la nube con la asistencia de Google Colaboratory, las imágenes y sus etiquetas fueron exportadas a Google Drive para su almacenamiento, posterior a ello, la compatibilidad de servicios provistos por el mismo fabricante permite una fácil exportación de los datos para proceder a realizar el entrenamiento.

3.2.2 Entrenamiento de YOLO

En la fase de entrenamiento de cualquier modelo de DL es necesario ajustar ciertos hiperparámetros, ya que de estos depende el funcionamiento y rendimiento de los algoritmos. Los hiperparámetros son parámetros adicionales que se establecen antes de que el modelo comience el proceso de aprendizaje y no son aprendidos directamente del conjunto de datos. A diferencia de los parámetros del modelo, que se estiman a partir de los datos mediante técnicas de optimización, los hiperparámetros son decisiones de diseño que se deben tomar antes de entrenar el modelo. Los más comunes son el tamaño de la red, el número de iteraciones (*epoch*), la resolución de entrada de las imágenes, el número de capas de la red, el número de clases, función de optimización, optimizador, etc.

En el caso de YOLO, su desempeño también depende del ajuste adecuado de sus hiperparámetros. Estos hiperparámetros incluyen factores como el tamaño de la red, el umbral de confianza, el tamaño de las cajas de detección y la resolución de entrada, entre otros. Al ajustar estos factores de manera eficiente, se pueden lograr detecciones más precisas y un equilibrio adecuado entre la velocidad de procesamiento y la precisión del modelo. En este sentido, explorar los hiperparámetros de YOLO se convierte en una tarea fundamental para obtener resultados satisfactorios en la detección de objetos (Jiang, 2021; Lakshmanan et al., 2021).

Dado que el objetivo de este trabajo es realizar una comparación de los optimizadores sobre la función de costo es importante analizar el hiperparámetro denominado tasa de aprendizaje. La tasa de aprendizaje determina la magnitud de los ajustes realizados a los pesos de las conexiones entre las neuronas durante el proceso de entrenamiento, es decir, es un valor numérico que controla la rapidez con la que el modelo aprende de los datos y actualiza los pesos en función del error calculado (Xiao et al., 2019). Así, una tasa de aprendizaje alta permite que el modelo converja más rápidamente durante el entrenamiento, pero también puede causar que se salte mínimos locales y no logre encontrar el mínimo global óptimo. Por otro lado, una tasa de aprendizaje baja puede hacer que el proceso de entrenamiento sea más lento, pero también permite una búsqueda más precisa y cuidadosa del

mínimo global. La elección de la tasa de aprendizaje se puede estimar considerando otros factores como el tamaño del set de datos, la complejidad del modelo y el ruido en los datos de entrenamiento.

Una vez seleccionada una tasa de aprendizaje adecuada se debe determinar el algoritmo optimizador usado en el entrenamiento. Los optimizadores utilizan diferentes técnicas para buscar eficientemente los valores óptimos de los parámetros. Algunos ejemplos de optimizadores populares son el descenso de gradiente estocástico (SGD, por sus siglas en inglés), el algoritmo de Adam, el RMSProp y el Adagrad, entre otros. Cada optimizador tiene su propio enfoque y variaciones en cómo actualizan los parámetros en función del gradiente. En este caso, para llevar a cabo la comparación se seleccionaron tres algoritmos muy utilizados en la literatura: SGD, Adam y AdamW.

El algoritmo optimizador SGD (Descenso del Gradiente Estocástico) realiza las actualizaciones de los parámetros mediante la aplicación de un subconjunto aleatorio de ejemplos de entrenamiento en cada iteración (Goodfellow et al., 2018; Mandt et al., 2017). El algoritmo se describe en la Ecuación 1.

$$w := w - \eta \nabla Q_i(w) \tag{1}$$

Dónde $\mathrm{Q(w)}$ es el gradiente de los cuadros delimitadores verdaderos.

En el optimizador SGD se debe establecer un hiperparámetro llamado *Momentum*, el cual, permite utilizar el promedio de los lotes previos del descenso del gradiente para que la red encuentre el mínimo rápidamente (Pomerat et al., 2019).

De acuerdo a los autores (Ariff & Ismail, 2023; Goodfellow et al., 2016; Kingma & Ba, 2015) el algoritmo optimizador Adam emplea las bondades del algoritmo SGD, pero añade un hiperparámetro llamado adaptador de la tasa de aprendizaje. Mediante la estimación adaptativa se determina el gradiente de la función de pérdida. Las ecuaciones 2, 3, 4, 5, 6 muestran el algoritmo representado matemáticamente.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (3)$$

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \quad (4)$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t} \quad (5)$$

$$\theta_t = \theta_{t+1} - \alpha \left(\frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \varepsilon} \right) \quad (6)$$

Donde:

g_t: Gradiente de la función de costo con respecto a los parámetros del modelo al instante t.

m_t: Estimación del *momentum* al instante t.

v_t : Estimación del *momentum* al instante t.

β_1 y β_2: Hiperparámetros que controlan la tasa de aprendizaje adaptativa y el *momentum*.

α: Tasa de aprendizaje.

AdamW presenta el mismo algoritmo de Adam, la diferencia radica en la aplicación de un factor de decaimiento de peso que permite obtener un rendimiento generalizable y estable de la red. El decaimiento de peso se aplica directamente a los pesos de cada actualización (Loshchilov & Hutter, 2019). La ecuación 7 indica la modificación final que presenta AdamW con respecto a Adam (Llugsi et al., 2021).

$$\theta_t = \theta_{t+1} - \alpha \left(\frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \varepsilon} + \lambda \theta_{t-1} \right) \quad (7)$$

Donde:

λ: Hiperparámetro de decaimiento de peso.

Después de definir los algoritmos optimizadores de función de coste se establece una tasa de aprendizaje de 0.001. De igual manera, de acuerdo con el optimizador que lo requiera se establecen los valores de *momentum* para SGD, β_1 y β_2 para Adam y adicional el hiperparámetro λ para AdamW. Estos valores se seleccionan de acuerdo a las recomendaciones de (Kingma & Ba, 2015; Konar et al., 2020) que coinciden con algunos valores que YOLOv5 mantiene por defecto. La Tabla 2 muestra la configuración y la selección de los hiperparámetros para cada optimizador.

Optimizador	**Tasa de aprendizaje**	**Momentum**	**Estimador Adaptativo**	**Decaimiento de peso**
SGD	0.001	0.937	No aplica	No aplica
Adam	0.001	0.937	0.99	No aplica
AdamW	0.001	0.937	0.99	0.0005

Tabla 2 Hiperparámetros para cada optimizador.

Adicionalmente, se configuran otros hiperparámetros de YOLOv5 que se resumen en la Tabla 3. Esta configuración coincide para los tres diferentes optimizadores.

Optimizador	**Modelo**	**Tamaño de imagen (px)**	**Semilla de aleatoriedad**	***Batch size***	**Épocas**
Adam	YOLOv5x	614 x 614	42	16	100
SGD	YOLOv5x	614 x 614	42	16	100
AdamW	YOLOv5x	614 x 614	42	16	100

Tabla 3 Configuración de los entrenamientos

La investigación de (Redmon et al., 2016) afirma que, el máximo valor obtenido en las épocas de entrenamiento de la red para las diferentes métricas, representa el

punto de mayor desempeño de la red, tal indicador se toma como el valor de referencia para los diferentes indicadores de rendimiento de YOLO.

Para determinar el rendimiento de la red se utilizan varias métricas como pueden ser la precisión, el *recall*, el índice de Jaccard, el mAP, el puntaje F1, la matriz de confusión y la pérdida (Hasan et al., 2022). De las cuales, se evaluaron la precisión, el recall, el mAP, la pérdida de cuadro y la pérdida de objetividad, con el fin de esclarecer diferencias significativas entre los diferentes optimizadores. La precisión evalúa la proporción de los ejemplos positivos que han sido clasificados como positivos correctamente (Manurung et al., 2022). El *recall* evalúa la capacidad de identificar correctamente los ejemplos positivos en los datos de validación (Manurung et al., 2022). Por otra parte, la métrica denominada *mean average precision* (mAP) toma en cuenta la media de las precisiones promedio (AP) y el número de clases *n* como se muestra en la Ecuación 8 y en la Ecuación 9.

$$AP = \int_0^1 PdR \tag{8}$$

$$mAP = \frac{\sum_{i=0}^{n} AP_i}{n} \tag{9}$$

3.2.3 Análisis de Resultados

Después de efectuar el entrenamiento en cada iteración, el modelo es evaluado en el conjunto de datos de validación. Con los datos de validación, la red es capaz de realizar las predicciones y comparar sus resultados con las etiquetas reales del *set* de validación, lo que permite calcular las métricas de rendimiento (Goodfellow et al., 2016). Las métricas de rendimiento se generan en cada época de entrenamiento de la red. Por tanto, para cada configuración de red con cada optimizador se obtienen cien ejemplos que servirán para realizar el análisis comparativo.

La aplicación del diseño factorial permite estudiar de manera simultánea los resultados de los factores provistos por los indicadores de rendimiento de la CNN que se establecieron con antelación. Por otra parte, se considera como la variable respuesta a los optimizadores. Mediante el uso del análisis estadístico, se determinaron diferencias significativas según las distintas métricas de evaluación. Por la variabilidad estocástica de los optimizadores y la reducción del sesgo, se efectuaron tres entrenamientos de YOLOv5 con cada optimizador.

Posteriormente, se procede a analizar los datos para determinar cómo se comparan los optimizadores en función de las diferentes métricas. Para lo cual se plantean pruebas de hipótesis y análisis de varianza para evaluar las diferencias, cada prueba a realizar con un nivel de significancia de 0.05.

Previo a realizar el diseño experimental, se debe comprobar si se cumplen las premisas y condiciones fundamentes que rigen los modelos estadísticos. Por lo cual, se realiza la prueba de Shapiro-Wilk para la normalidad y para la homocedasticidad la prueba de Levene. En función de no aceptar la hipótesis nula en la prueba de normalidad, se utilizaron técnicas de estadística no paramétrica como la prueba H de Kruskal- Wallis que permite corroborar si existen diferencias significativas entre los algoritmos optimizadores.

También se realiza pruebas *post hoc*, considerando las técnicas de estadística no paramétrica necesarias en la ejecución del análisis, la prueba de Dunn que permite realizar las comparaciones múltiples según el nivel de significancia de 0.05.

4. Resultados y Discusión

4.1 Pérdida en los conjuntos de entrenamiento y validación de YOLO

Es importante conocer cómo evolucionó la pérdida de caja al momento de realizar los entrenamientos con cada optimizador, debido a que permite evidenciar el progreso del modelo en el entrenamiento, y cómo se ajustan los datos al mismo, la idea fundamental es conseguir que la pérdida se minimice. La Figura 2 muestra la pérdida de caja para los optimizadores Adam, SGD y AdamW respectivamente.

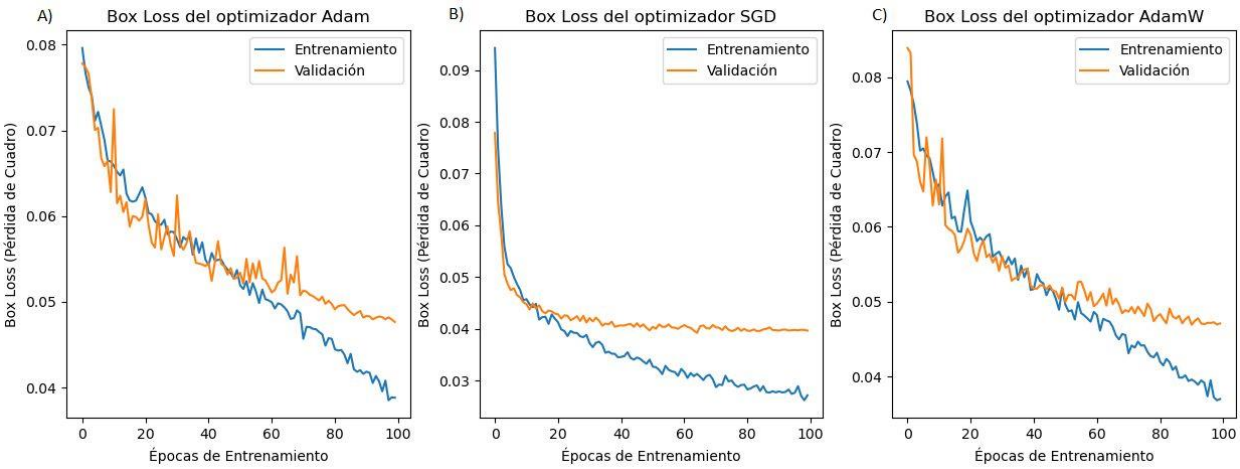

Figura 2 Comparativa de Box Loss entre los optimizadores

De las imágenes se puede evidenciar un descenso pronunciado de la pérdida tanto para entrenamiento como para la validación, fundamentándose en la evidencia, se puede afirmar que la red está aprendiendo de los datos, a medida que avanzan las iteraciones de los entrenamientos, la pérdida de cuadro disminuye. La Figura 2(B) presenta la pérdida de cuadro de la red entrenada con SGD como optimizador, claramente se puede notar que la curva es suave, lo que indica que el modelo está realizando ajustes graduales y generalizando de manera correcta a una solución óptima. Por otra parte, los optimizadores Adam y AdamW presentan datos fluctuantes en su rendimiento. Se debe destacar que las mediciones de los tres algoritmos optimizadores rondan límites similares del mínimo y máximo valor en las

100 épocas de entrenamiento, no obstante, es claro que en las tres gráficas la curva de validación tiende a superar a la de entrenamiento, lo que indica que el modelo está presentando un sobreajuste.

Al igual que la pérdida de cuadro, parte perteneciente a la función de coste es la pérdida de objetividad, se debe considerar la evolución de la pérdida de objetividad a través de las épocas de entrenamiento. Por ello, la Figura 3 presentan la pérdida de objetividad para los entrenamientos con los optimizadores Adam, SGD y AdamW respectivamente.

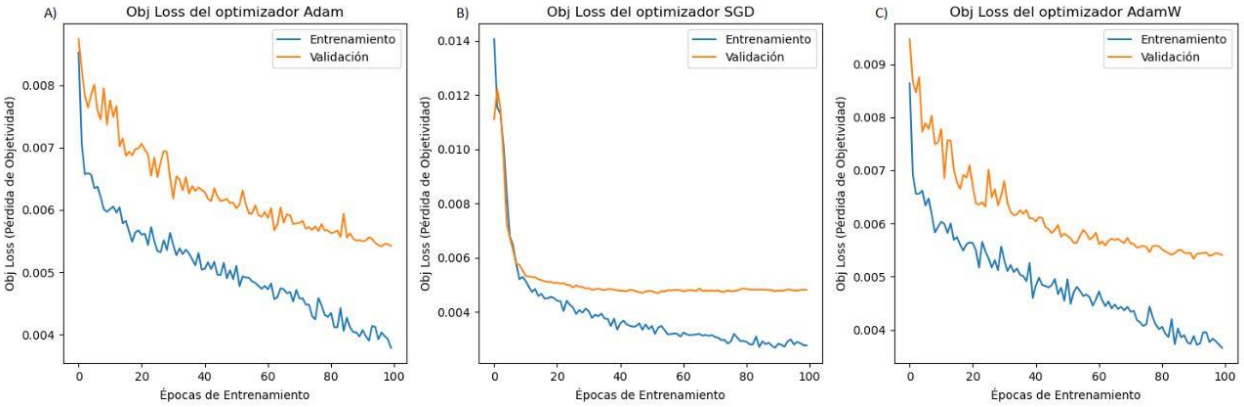

Figura 3 Comparativa de Obj Loss entre los optimizadores

Al evidenciar la Figura 3, se puede determinar que, a través de las diferentes épocas de entrenamiento, el modelo aprende de los datos ya que se tiene una disminución en la pérdida de objetividad de los diferentes optimizadores. Es notorio que el optimizador SGD presenta un descenso más pronunciado y a la vez, una curva más suave en los valores de la pérdida de objetividad, sin olvidar que en el caso de SGD, el valor máximo donde inician las curvas es notoriamente superior en comparación con los optimizadores Adam y AdamW. Al igual que con la pérdida de cuadro, las gráficas de pérdida de objetividad de Adam y AdamW muestran fluctuaciones, lo que significa que con estos optimizadores la red tiene problemas para buscar la convergencia en las primeras épocas de entrenamiento. Se debe destacar que las

gráficas de pérdida de objetividad de validación presentan valores mayores a los de entrenamiento en los tres optimizadores, esto sugiere que el modelo es muy complicado para la generalización de nuevos datos o que existen problemas de sobreajuste.

4.2 Métricas del rendimiento en el conjunto de validación de YOLO.

La Figura 4 presenta la comparativa de las métricas de precisión, *recall* y mAP entre cada uno de los optimizadores.

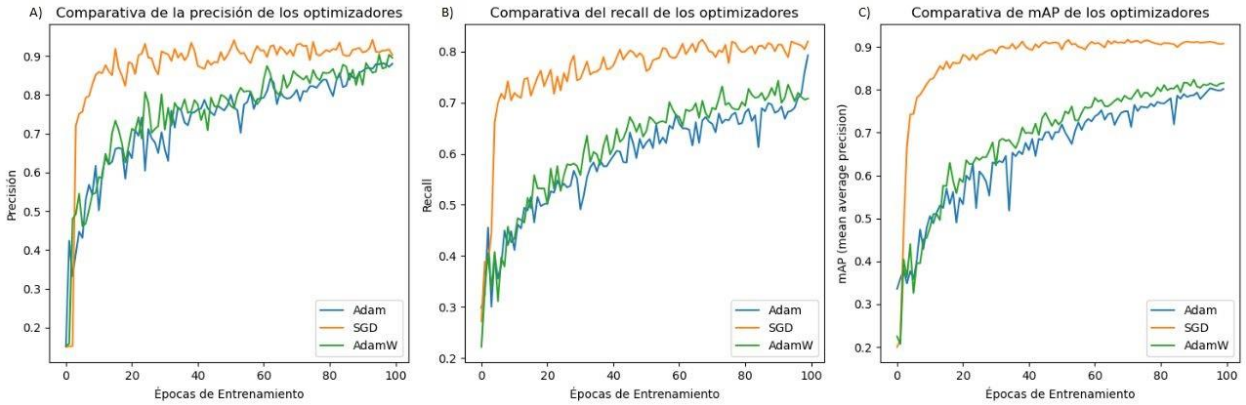

Figura 4 comparativa de las métricas de precisión, *recall* y mAP entre cada uno de los optimizadores

La Figura 4(A) muestra la evolución de la precisión de la red con cada uno de los optimizadores. La evidencia indica que con el optimizador SGD se alcanzan los valores más altos de precisión para la red. Además, es notorio el comportamiento de la precisión con el optimizador SGD, ya que alcanza los valores más altos en menos épocas de entrenamiento. Los optimizadores Adam y AdamW muestran un crecimiento similar en la búsqueda de su mejor valor de precisión. Es importante destacar que las curvas presentan fluctuaciones, esto puede ser causado por la tendencia al sobreajuste del modelo evidenciado en las pérdidas o también por el tamaño del lote configurado para los entrenamientos.

La Figura 4(B) presenta la comparativa de la métrica del *recall* con los tres optimizadores estudiados. Es evidente que el optimizador SGD alcanza los valores más altos de *recall* incluso en menor cantidad de épocas de entrenamiento. La gráfica muestra que el ascenso y la búsqueda del valor máximo de Adam y AdamW es bastante similar. De la ilustración se puede notar que los algoritmos optimizadores presentan dificultad en establecer su punto de máximo *recall*, ya que las gráficas presentar picos leves de ascenso y descenso.

La Figura 4(C) muestra la comparativa de la métrica *mean average precision* para los tres optimizadores. Al igual que con las métricas anteriores, el entrenamiento con el optimizador SGD alcanza valores superiores de mAP en menor cantidad de épocas de entrenamiento. Se debe destacar que la curva del optimizador SGD es estable y no presenta mucha dispersión como el caso de los entrenamientos de Adam y AdamW, estos últimos muestran comportamientos muy similares y se puede observar picos leves en sus respectivas gráficas.

4.3 Detección de peatones.

La Figura 5 contempla la detección de los peatones dada por los tres optimizadores y el porcentaje de confianza que tiene cada uno de ellos para determinar si el objeto pertenece a la clase peatón o no. En la ilustración, el par de imágenes mostrado en a), presenta la detección de peatones con el optimizador Adam; b) muestra la detección con el optimizador SGD y c) con AdamW.

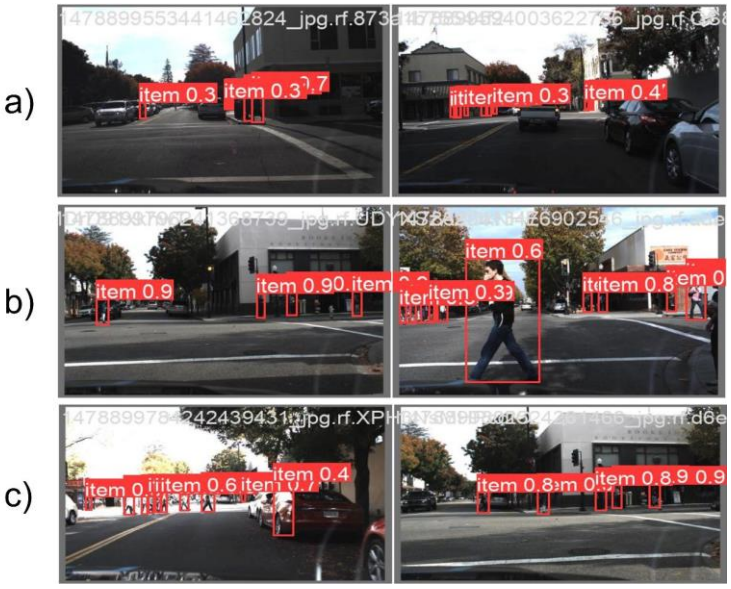

Figura 5 Detección de peatones con los tres optimizadores

Las etiquetas de las imágenes contienen la confianza de la red al detectar los objetos de la clase, como se mencionó, el par de imágenes asociadas a la Figura 5(b) muestra la confianza desarrollada por la red al ser entrenada con el optimizador SGD, se puede observar que SGD presenta la mayor cantidad de detecciones que sobrepasa el umbral del 80% de confianza al momento de la detección, así mismo, AdamW muestra excelentes resultados al momento de determinar si un objeto pertenece a la clase peatón. En contraste con Adam que muestra una confianza en la detección del 30% y 40%.

4.4 Diseño Experimental.

Como se mencionó en la sección 3 correspondiente a la metodología, se comprobó los supuestos de modelo para escoger la metodología adecuada para realizar las comprobaciones múltiples, y en base a los factores establecidos se puede determinar su incidencia en la elección del mejor algoritmo optimizador para entrenar la red neuronal YOLO del experimento. En primera instancia, se realizó la prueba de normalidad y luego la prueba de homocedasticidad. En el Anexo 1 se encuentra un

enlace que lleva al repositorio de GitHub donde se puede evidenciar el desarrollo de las diferentes pruebas.

4.4.1 Prueba de normalidad

La prueba escogida para comprobar el supuesto de normalidad fue la prueba de Shapiro-Wilk, con ella se puede evaluar la hipótesis nula de que los valores de los datos de los factores siguen una distribución normal. Se debe tener en cuenta que la prueba de normalidad se aplica a cada métrica por separado para obtener información más detallada individualmente.

Optimizador	**Métrica**	**Estadístico de Prueba**	**P-valor**
Adam	Precisión	0.8472	1.47e-16
Adam	Recall	0.9211	1.77e-11
Adam	mAP	0.8838	2.50e-14
Adam	Train/Box Loss	0.9677	3.0e-6
Adam	Val/Box Loss	0.8385	5.01e-17
Adam	Train/Obj Loss	0.9388	8.31e-10
Adam	Val/Obj Loss	0.8914	8.33e-14
SGD	Precisión	0.3714	6.97e-31
SGD	Recall	0.5359	2.06e-27
SGD	mAP	0.3859	1.30e-30
SGD	Train/Box Loss	0.7260	6.76e-22
SGD	Val/Box Loss	0.4651	5.21e-29
SGD	Train/Obj Loss	0.5846	3.37e-26
SGD	Val/Obj Loss	0.3527	3.12e-31
AdamW	Precisión	0.7759	5.73e-20
AdamW	Recall	0.8567	5.12e-16
AdamW	mAP	0.8244	9.37e-18
AdamW	Train/Box Loss	0.9493	1.17e-08

AdamW	Val/Box Loss	0.7928	3.05e-19
AdamW	Train/Obj Loss	0.9256	4.48e-11
AdamW	Val/Obj Loss	0.8110	2.08e-18

Tabla 4 Resultados prueba de Shapiro-Wilk

La Tabla 4 muestra los resultados obtenidos de la prueba de normalidad de Shapiro-Wilk, los resultados de los p valores para cada prueba al ser menores que el nivel de significancia planteado, sugieren que se debe rechazar la hipótesis nula, en consecuencia, se indica que los datos de los factores no siguen una distribución normal. Al no cumplirse el supuesto de normalidad, las técnicas de estadística paramétrica para comprobaciones múltiples no son las adecuadas, por ende, se procede a utilizar la prueba de Kruskal-Wallis en lugar del ANOVA para las comparaciones múltiples.

4.4.2 Prueba de homocedasticidad

La prueba de homocedasticidad permite comprobar las varianzas de los diferentes factores, estableciendo como su hipótesis nula de que sus varianzas son iguales. La prueba escogida para la homocedasticidad es la prueba de Levene. Para aplicar la prueba de Levene, se evalúa por separado cada métrica perteneciente a los tres optimizadores del experimento.

Métrica	**Estadístico de Prueba**	**P-valor**
Precisión	3.6033	3.93e-4
Recall	5.7423	3.77e-7
mAP	8.0880	1.39e-10
Train/Box Loss	1.3589	2.11e-1
Val/Box Loss	6.8185	1.02e-8
Train/Obj Loss	2.1002	3.3e-2
Val/ Obj Loss	1.1214	3.45e-1

Tabla 5 Resultados de la prueba de Levene para homocedasticidad

De la Tabla 5 se evidencia que el supuesto de homocedasticidad únicamente acepta la hipótesis nula en las métricas de Train/Box Loss, Train/Obj Loss y Val/Obj Loss. Las demás métricas muestran un p valor menor al nivel de significancia establecido, por ende, existe suficiente evidencia para descartar la hipótesis nula.

4.4.3 Comparaciones Múltiples

Al comprobarse el incumplimiento de los supuestos de modelo, se optó por utilizar metodologías de estadística no paramétrica. De acuerdo con la investigación, la prueba de Kruskal-Wallis fue seleccionada para realizar las comparaciones múltiples. La prueba sugiere ser aplicada para cada métrica por separado con la finalidad de comparar los grupos correspondientes a los optimizadores. La prueba de Kruskal-Wallis evalúa la hipótesis nula de que las distribuciones de los grupos son iguales.

Métrica	**Estadístico de Prueba**	**P-valor**
Precisión	408.311	2.16e-89
Recall	487.35	1.48e-16
mAP	496.35	1.65e-108
Train/Box Loss	418.22	1.52e-91
Val/Box Loss	529.76	9.20e-116
Train/Obj Loss	286.82	5.20e-63
Val/ Obj Loss	445.13	2.18e-97

Tabla 6 Resultados de la prueba de Kruskal-Wallis para comparaciones múltiples

Como se muestra en la Tabla 6, valores cercanos a cero del p valor, indica que entre los diversos algoritmos optimizadores existen grandes diferencias entre las observaciones de sus métricas. Los valores del estadístico H de prueba al ser bastante altos confirman este supuesto. Los valores extremadamente cercanos a cero se asocian a la aparición de valores atípicos en los datos del ensayo, esto se puede

evidenciar en los diagramas de caja contenidos en el código del Anexo 1. Para determinar específicamente cuales algoritmos difieren entre sí, es necesario el uso de comparaciones post hoc.

4.4.4 Comparaciones *post hoc*

Se estableció que existe diferencias significativas con la prueba de Kruskal-Wallis en las métricas de evaluación de los tres optimizadores en cuestión. Para determinar exclusivamente los grupos que difieren entre sí, es necesaria la comparación de rangos implementada por la prueba post hoc de Dunn, una prueba no paramétrica generalmente usada después de efectuar la prueba de Kruskal-Wallis. Se aplicó el método de corrección de múltiples comparaciones de Bonferroni a la prueba de Dunn.

Precisión			
Optimizadores	Adam	AdamW	SGD
Adam	1	9.6e-2	1.02e-75
AdamW	9.6e-2	1	1.84e-59
SGD	1.02e-75	1.84e-59	1

Tabla 7 Comparaciones de Dunn para la precisión

Recall			
Optimizadores	Adam	AdamW	SGD
Adam	1	9.58e-3	3.24e-92
AdamW	9.58e-3	1	6.90e-68
SGD	3.24e-92	6.90e-68	1

Tabla 8 Comparaciones de Dunn para el Recall

mAP			
Optimizadores	Adam	AdamW	SGD
Adam	1	1.03e-2	1.03e-93
AdamW	1.038e-2	1	2.31e-69
SGD	1.03e-93	2.31e-69	1

Tabla 9 Comparaciones de Dunn para mAP

Train Box Loss			
Optimizadores	Adam	AdamW	SGD
Adam	1	9.89e-2	2.14e-77
AdamW	9.89e-2	1	4.94e-61
SGD	2.14e-77	4.94e-61	1

Tabla 10 Comparaciones de Dunn para Train Box Loss

Val Box Loss			
Optimizadores	Adam	AdamW	SGD
Adam	1	1.13e-2	1.79e-99
AdamW	1.13e-2	1	1.50e-74
SGD	1.79e-79	1.50e-74	1

Tabla 11 Comparaciones de Dunn para Val Box Loss

Train Obj Loss			
Optimizadores	Adam	AdamW	SGD
Adam	1	5.13e-2	3.68e-55
AdamW	5.13e-2	1	4.7e-40
SGD	3.68e-55	4.7e-40	1

Tabla 12 Comparaciones de Dunn para Train Obj Loss

Val Obj Loss			
Optimizadores	Adam	AdamW	SGD
Adam	1	4.82e-2	5.87e-83
AdamW	4.82e-2	1	5.86e-64
SGD	5.87e-83	5.86e-64	1

Tabla 13 Comparaciones de Dunn para Val Obj Loss

De la Tabla 7 hasta la Tabla 13 se puede apreciar la matriz de comparaciones por grupos para cada uno de los indicadores de rendimiento de la red neuronal convolucional. De cada una de las métricas es notoria la diferencia que presenta el optimizador SGD con respecto a Adam y AdamW. Esto debido a que presenta sus p valores muy por debajo del nivel de significancia establecido, con esto, queda demostrado estadísticamente lo que se puede apreciar en las gráficas de los indicadores mostradas anteriormente.

4.4.5 Discusión

La investigación de (Kumar & Kumar, 2023) realizó un análisis comparativo entre los optimizadores Adam y SGD con un *momentum* de 0.8, en redes YOLOv5 y YOLOv7 para la detección de objetivos militares. De manera similar, concluyeron que, para dicha aplicación el optimizador SGD presentó los mejores resultados en el rendimiento de la red, evaluando el rendimiento en las métricas de mAP y puntaje F1. En contraste a esta investigación, los investigadores aplicaron un *momentum* para SGD empezando en 0.9 y reduciéndolo a 0.8 donde encontraron el mejor rendimiento, en esta investigación, se aplicó un *momentum* de 0.937 y de igual manera, se demostró que SGD presentó el mejor rendimiento en métricas como mAP, precisión, *recall* e incluso en las diferentes pérdidas generadas por la red.

De acuerdo con el artículo de los autores (Seelwal & Sharma, 2022), YOLO no es la única arquitectura para la detección de objetos. En esta investigación, se propuso la construcción de una CNN propia para la detección y clasificación de enfermedades

en cultivos de arroz. A su vez, la investigación realizó un análisis comparativo entre los optimizadores SGD y ADAM y el rendimiento mostrado en diferentes clases de enfermedades de los cultivos. En esta investigación, se realizó una red de detección y clasificación multiclase, siendo la red entrenada con el optimizador SGD la que mejor desempeño muestra en métricas como la precisión, la exactitud, el *recall* y el puntaje F1, de esta manera, en similitud a los resultados obtenidos en esta investigación. Se debe destacar la importancia de la creación de un modelo de detección multiclase para la aplicación en sistemas de conducción autónoma, debido a que no solo los peatones son agentes pertenecientes a un escenario de conducción, para una aplicación más adecuada, se necesita un modelo que pueda realizar una detección multiclase en la conducción.

Este artículo propone la utilización de la versión 5 de YOLO en su modelo más complejo; sabiendo que la CNN cuenta con otras versiones, se considera importante realizar el análisis comparativo mediante la implementación de diferentes versiones de la red. La investigación desarrollada por (Muksit et al., 2022) muestra la implementación de un detector de fauna marina basado en la versión 3, la versión 4 y en la propuesta de dos modelos creados por los investigadores llamados YOLO-Fish-1 y YOLO-Fish-2. Se denota importante, ya que otras versiones de la arquitectura de YOLO y la propuesta de modelos basados en la misma pueden mejorar significativamente el rendimiento de la red, por tal motivo, la red propuesta por los investigadores YOLO-Fish-1 obtuvo el mejor desempeño en comparación con las demás. Fundamentándose en esta premisa, se debe considerar importante la implementación de una red de detección de peatones basada en diferentes versiones de YOLO, o como propone el autor (Xue et al., 2021), la creación de un detector de peatones basado en la arquitectura de YOLO.

Cuando se desea aplicar una red neuronal a un proyecto tangible, es importante evaluar el desempeño de la red mediante un *set* de datos de prueba, de acuerdo con el artículo propuesto por (Ma et al., 2022) aplicaron un modelo YOLOv3, YOLOv5 y YOLO personalizado en la construcción de un dispositivo detector de cigarrillos. Los investigadores decidieron dividir el conjunto de datos en entrenamiento, validación

y prueba, considerando que el modelo será aplicado a una situación práctica y no con fines investigativos. Se debe denotar que utilizaron un conjunto de prueba debido a que el conjunto de validación de la investigación es relativamente pequeño. En contraste, esta investigación utilizó un conjunto de validación de 681 imágenes, por lo cual, se considera que existe una gran generalización de datos no vistos en el modelo entrenado.

5. Conclusiones

Este trabajo realiza la comparación del desempeño de una red neuronal convolucional YOLOv5 aplicada a la detección de peatones usando tres diferentes optimizadores en el entrenamiento. El resultado de la prueba de normalidad y homocedasticidad permitió determinar el incumplimiento de los supuestos del modelo, con ello, se procedió a utilizar pruebas de estadística no paramétrica para establecer estadísticamente diferencias significativas.

De las pruebas de comparación múltiple se deduce que existen diferencias significativas entre los tres algoritmos del experimento, basado en el resultado, se realizó la prueba *post hoc* de Dunn y muestra que el modelo entrenado con el optimizador SGD presentó mejores resultados en comparación con los optimizadores Adam y AdamW. En cuanto al rendimiento general de la red, los valores más altos los consigue SGD con 95% de precisión, 82% *recall* y un 91% mAP.

Es importante destacar que la gran mayoría de los modelos de aprendizaje profundo no son generalizables, con esta mención, se afirma que específicamente para la versión de YOLOv5 con su modelo *X-Large* entrenado con el conjunto de imágenes Udacity Self Driving Car y construido para la detección de una sola clase. Los resultados indican que el algoritmo optimizador de función de pérdida SGD se considera el mejor para este problema de aplicación.

6. Agradecimientos

Muestro un profundo agradecimiento y aprecio a la Magister Paulina Morillo, quien afablemente decidió aceptar la tutoría de esta investigación y además con su paciencia, sus conocimientos y su guía, aportó de valor incalculable en el desarrollo y la culminación de este proyecto.

Referencias

Ariff, N. A. M., & Ismail, A. R. (2023). Study of Adam and Adamax Optimizers on AlexNet Architecture for Voice Biometric Authentication System. *2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 1–4. https://doi.org/10.1109/IMCOM56909.2023.10035592

Campesato, O. (2020). *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning & Information. http://ebookcentral.proquest.com/lib/upsal/detail.action?docID=6032875

Ge, L., & Dou, L. (2023). G-Loss: A loss function with gradient information for super-resolution. *Optik*, *280*, 170750. https://doi.org/10.1016/j.ijleo.2023.170750

Goodfellow, I., Bengio, Y., & Aaron, C. (2018). Deep learning. *Genetic Programming and Evolvable Machines*, *19*(1–2), 305–307. https://doi.org/10.1007/s10710-017-9314-z

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://books.google.com.ec/books?id=Np9SDQAAQBAJ

Gu, W., & Sun, K. (2023). AYOLOv5: Improved YOLOv5 based on attention mechanism for blood cell detection. *Biomedical Signal Processing and Control*, 105034. https://doi.org/10.1016/j.bspc.2023.105034

Hasan, M. J., Wan Ahmad, W. S. H. M., Fauzi, M. F. A., Lee, J. T. H., Khor, S. Y., Looi, L. M., & Abas, F. S. (2022). Nuclei Segmentation in ER-IHC Stained Histopathology Images using Mask R-CNN. *2022 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, 1–4. https://doi.org/10.1109/ISPACS57703.2022.10082832

Huang, R., Gu, J., Sun, X., Hou, Y., & Uddin, S. (2019). A Rapid Recognition Method for Electronic Components Based on the Improved YOLO-V3 Network. *Electronics*, *8*(8), 825. https://doi.org/10.3390/electronics8080825

Jiang, H. (2021). Machine learning fundamentals : A concise introduction. In *Cambridge University Press*.

Kaszas, D., & Roberts, A. C. (2022). Comfort with varying levels of human supervision in

self-driving cars: Determining factors in Europe. *International Journal of Transportation Science and Technology*. https://doi.org/10.1016/j.ijtst.2022.08.001

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Konar, J., Khandelwal, P., & Tripathi, R. (2020). Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network. *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 1–5. https://doi.org/10.1109/SCEECS48394.2020.94

Kumar, S., & Kumar, C. (2023). Deep Learning based Target detection and Recognition using YOLO V5 algorithms from UAVs surveillance feeds. *2023 International Conference for Advancement in Technology (ICONAT)*, 1–5. https://doi.org/10.1109/ICONAT57137.2023.10080677

Lakshmanan, V., Görner, M., & Gillard, R. (2021). *Practical Machine Learning for Computer Vision End-to-End Machine Learning for Images* (L. Piper Editorial Consulting (ed.); First Edit). O'Reilly Media, Inc.

Li, Y., Cheng, R., Zhang, C., Chen, M., Ma, J., & Shi, X. (2022). Sign language letters recognition model based on improved YOLOv5. *2022 9th International Conference on Digital Home (ICDH)*, 188–193. https://doi.org/10.1109/ICDH57206.2022.00036

Llugsi, R., Yacoubi, S. El, Fontaine, A., & Lupera, P. (2021). Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito. *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, 1–6. https://doi.org/10.1109/ETCM53643.2021.9590681

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*, *100*.

Ma, Y., Yang, J., Li, Z., & Ma, Z. (2022). YOLO-Cigarette: An effective YOLO Network for outdoor smoking Real-time Object Detection. *2021 Ninth International Conference on Advanced Cloud and Big Data (CBD)*, 121–126. https://doi.org/10.1109/CBD54617.2021.00029

Mandt, S., Hof Fman, M. D., & Blei, D. M. (2017). Stochastic gradient descent as approximate Bayesian inference. *Journal of Machine Learning Research*, *18*, 1–35.

Manurung, P., Rusmin, P. H., & Yusuf, R. (2022). Custom Pallet Detection Using YOLOv5 Deep Learning Architecture. *2022 International Symposium on Electronics and Smart Devices (ISESD)*, 1–6. https://doi.org/10.1109/ISESD56103.2022.9980635

Mathew, A., Amudha, P., & Sivakumari, S. (2021). *Deep Learning Techniques: An Overview* (pp. 599–608). https://doi.org/10.1007/978-981-15-3383-9_54

Mathew, M. P., & Mahesh, T. Y. (2022). Leaf-based disease detection in bell pepper plant using YOLO v5. *Signal, Image and Video Processing*, *16*(3), 841–847. https://doi.org/10.1007/s11760-021-02024-y

Meena, S. D., Gayathri siva sameeraja, V., Lasya, N. S., Sathvika, M., Harshitha, V., & Sheela, J. (2022). Hybrid Neural Network Architecture for Multi-Label Object Recognition using Feature Fusion. *Procedia Computer Science*, *215*, 78–90. https://doi.org/10.1016/j.procs.2022.12.009

Muksit, A. Al, Hasan, F., Hasan Bhuiyan Emon, M. F., Haque, M. R., Anwary, A. R., & Shatabda, S. (2022). YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment. *Ecological Informatics*, *72*, 101847. https://doi.org/10.1016/j.ecoinf.2022.101847

Pathak, P., Gangwar, H., & Jalal, A. S. (2020). Performance Analysis of Gradient Descent Methods for Classification of Oranges using Deep Neural Network. *2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)*, 68–72. https://doi.org/10.23919/INDIACom49435.2020.9083723

Pérez González, R. (2021). *Deep Learning Methods for Automotive Radar Signal Processing*. Cuvillier Verlag. http://ebookcentral.proquest.com/lib/upsal/detail.action?docID=6685229

Perumal, P. S., Sujasree, M., Chavhan, S., Gupta, D., Mukthineni, V., Shimgekar, S. R., Khanna, A., & Fortino, G. (2021). An insight into crash avoidance and overtaking advice systems for Autonomous Vehicles: A review, challenges and solutions. *Engineering Applications of Artificial Intelligence*, *104*, 104406. https://doi.org/10.1016/j.engappai.2021.104406

Pomerat, J., Segev, A., & Datta, R. (2019). On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression. *2019 IEEE International*

Conference on Big Data (Big Data), 6183–6185. https://doi.org/10.1109/BigData47090.2019.9005674

Rahman, F., Ritun, I. J., Farhin, N., & Uddin, J. (2019). An assistive model for visually impaired people using YOLO and MTCNN. *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, 225–230. https://doi.org/10.1145/3309074.3309114

Ranjan, S., & Senthamilarasu, S. (2020). *Applied deep learning and computer vision for self-driving cars*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Seelwal, P., & Sharma, A. (2022). Automatic Detection of Rice Diseases using Deep Convolutional Neural Networks with SGD and ADAM. *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 1256–1260. https://doi.org/10.1109/ICAC3N56670.2022.10074538

Shabrina, N. H., Lika, R. A., & Indarti, S. (2023). Deep learning models for automatic identification of plant-parasitic nematode. *Artificial Intelligence in Agriculture*, *7*, 1–12. https://doi.org/10.1016/j.aiia.2022.12.002

Sudars, K., Namatevs, I., Judvaitis, J., Balass, R., Nikulins, A., Peter, A., Strautina, S., Kaufmane, E., & Kalnina, I. (2022). YOLOv5 Deep Neural Network for Quince and Raspberry Detection on RGB Images. *2022 Workshop on Microwave Theory and Techniques in Wireless Communications (MTTW)*, 19–22. https://doi.org/10.1109/MTTW56973.2022.9942550

Udacity. (2018). *Open Source Self-Driving Cars*. https://github.com/udacity/self-driving-car

Ultralytics. (2022). *YOLOv5: The friendliest AI architecture you'll ever use*. https://ultralytics.com/yolov5

Vani, S., & Rao, T. V. M. (2019). An Experimental Approach towards the Performance Assessment of Various Optimizers on Convolutional Neural Network. *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 331–336. https://doi.org/10.1109/ICOEI.2019.8862686

Verma, S., Tripathi, S., Singh, A., Ojha, M., & Saxena, R. R. (2021). Insect Detection and Identification using YOLO Algorithms on Soybean Crop. *TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON)*, 272–277. https://doi.org/10.1109/TENCON54134.2021.9707354

Xue, Y., Ju, Z., Li, Y., & Zhang, W. (2021). MAF-YOLO: Multi-modal attention fusion based YOLO for pedestrian detection. *Infrared Physics & Technology*, *118*, 103906. https://doi.org/10.1016/j.infrared.2021.103906

Zhang, J., Chen, X., Li, Y., Chen, T., & Mou, L. (2021). Pedestrian detection algorithm based on improved Yolo v3. *2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, 180–183. https://doi.org/10.1109/ICPICS52425.2021.9524267

Anexos

Anexo 1

https://github.com/tavomx45/trabajoDeTitulacion.git