



# Detecting and Classifying Traffic Signs Using Deep Learning

Raúl Andrino, Antonio Reviejo

*Universidad Politécnica de Madrid. Escuela Técnica Superior de Ingeniería de Sistemas Informáticos*

## Resumen

*La clasificación automática de señales de tráfico es una tarea crucial en sistemas de conducción autónoma y de asistencia al conductor. En este trabajo, se aborda la detección y posteriormente clasificación de señales de tráfico utilizando técnicas de aprendizaje profundo. Primero se estudiarán y evaluarán diferentes modelos YOLO preentrenados para detectar señales de tráfico en la carretera. Y después se desarrollarán y evaluarán diferentes modelos de clasificación CNN para clasificar las señales en sus respectivos tipos, además de distintas técnicas de preprocessamiento de las imágenes que mejoren estos modelos. Para ello se eligió el conjunto de datos German Traffic Sign Recognition Benchmark (GTSRB), que incluye imágenes de señales de tráfico alemanas de 43 tipos distintos.*

*Finalmente, se probarán los modelos con un vídeo grabado desde el coche, que recorrerá los alrededores del Campus Sur de la Universidad Politécnica de Madrid.*

**Palabras Claves:** Clasificación, YOLO, CNN, Traffic Sign Detection, CLAHE

## 1. Introducción

El reconocimiento automático de señales de tráfico es un componente esencial en el desarrollo de sistemas de conducción autónoma y en las tecnologías avanzadas de asistencia al conductor (ADAS, por sus siglas en inglés). Las señales de tráfico proporcionan información crítica para los conductores y los vehículos, como límites de velocidad, advertencias de peligro y normativas viales, que deben ser interpretadas de manera precisa y en tiempo real para garantizar la seguridad vial. La capacidad de un vehículo de detectar y clasificar señales mientras está en movimiento es clave para reducir la dependencia de la intervención humana y mejorar la eficiencia del tránsito.

En este contexto, los avances recientes en aprendizaje profundo, especialmente en los modelos YOLO y las redes neuronales convolucionales (CNN), han revolucionado las tareas de visión por computador al permitir la detección y clasificación precisa de objetos en tiempo real. Sin embargo, este problema plantea múltiples desafíos: variaciones en las condiciones ambientales (iluminación, lluvia, nieve), señales parcialmente ocultas, desgaste por el tiempo, y diferencias en el diseño de señales según la región. Estos factores complican la implementación de sistemas robustos y confiables en escenarios reales.

El presente trabajo se centra en diseñar y evaluar

un modelo que sea capaz de detectar señales de tráfico mientras un vehículo está en movimiento, identificándolas de manera precisa y clasificándolas en tiempo real según su significado. Para abordar este problema, se utiliza el conjunto de datos *German Traffic Sign Recognition Benchmark* (GTSRB), conocido por su diversidad y complejidad, como base para el entrenamiento y la validación del modelo. A través de este enfoque, se busca desarrollar un sistema que no solo obtenga un alto nivel de precisión en un entorno controlado, sino que también sea capaz de adaptarse a las condiciones del mundo real.

El objetivo principal es implementar un modelo preentrenado YOLO y basado en redes neuronales convolucionales que, combinado con estrategias de preprocessamiento y optimización, permita detectar y clasificar señales de tráfico con la velocidad y precisión necesarias para su integración en vehículos en movimiento. Este trabajo representa un paso hacia la creación de sistemas inteligentes que contribuyan a la seguridad vial, sirviendo de base para investigaciones futuras y aplicaciones en entornos reales.

## 2. Estado del Arte

El reconocimiento de señales de tráfico es un campo que ha experimentado un avance significativo gracias a

la adopción de técnicas de aprendizaje profundo. Este problema se aborda principalmente en dos etapas: la detección de las señales en imágenes de escenas de tráfico y su clasificación en las categorías correspondientes. A continuación, se presenta un análisis detallado de las herramientas, técnicas y enfoques más recientes en este campo.

## 2.1. Modelos de detección de objetos

Los modelos de detección de objetos han evolucionado significativamente gracias al aprendizaje profundo. Estas herramientas permiten identificar señales de tráfico en tiempo real, incluso en condiciones desafiantes.

YOLO (*You Only Look Once*) es uno de los frameworks más populares y eficientes para la detección en tiempo real. Su principal ventaja radica en su arquitectura unificada, que permite procesar una imagen completa en una sola pasada, logrando una combinación de alta velocidad y precisión.

- **Versiones recientes:** YOLOv4 introdujo optimizaciones como *Mosaic data augmentation*, *CSPDarknet* y *Bag of Freebies*, mejorando tanto la precisión como la velocidad[1]. Posteriormente, YOLOv5, desarrollado por *Ultralytics*, incluyó implementaciones más accesibles y compatibles con *PyTorch*[8]. YOLOv8 ha destacado recientemente por integrar técnicas de aprendizaje sin anclas (*anchor-free learning*) y *knowledge distillation*, lo que aumenta la capacidad de generalización de los modelos en escenarios reales[18]. Para esta aplicación, se utilizará la versión 8 de este modelo.
- **Limitaciones y mejoras:** Aunque YOLO es robusto, enfrenta dificultades con objetos pequeños o superpuestos. Para mitigar estos problemas, se han propuesto modificaciones en la función de pérdida o la creación de submodelos especializados según los objetos a detectar.
- **Aplicaciones en señales de tráfico:** Estudios como [6] adaptaron YOLOv4 para la detección de señales en tiempo real, alcanzando una alta precisión en condiciones del entorno complejas. Además, combinaciones de YOLO con técnicas de mejora de imágenes, como CLAHE, han demostrado ser efectivas en entornos complejos[15].

Por otro lado, modelos como **DETR** (*Detection Transformer*) han comenzado a destacar en tareas de detección de objetos. A diferencia de los enfoques tradicionales basados en anclas, DETR utiliza un mecanismo de atención global para identificar objetos, lo que simplifica la arquitectura del modelo y mejora la robustez frente a variaciones ambientales.

Trabajos como [17] han aplicado DETR para detectar señales de tráfico en condiciones adversas, mostrando una mayor resiliencia frente a factores como la oclusión y el desgaste de las señales. Sin embargo, la

desventaja principal de DETR sigue siendo su mayor tiempo de inferencia en comparación con YOLO, lo que limita su uso en aplicaciones de tiempo real.

Faster R-CNN[12] sigue siendo una opción robusta para la detección de objetos en alta precisión. Aunque es más lento que YOLO, su capacidad para generar *region proposals* más precisas ha sido útil en escenarios donde la velocidad no es prioritaria. Investigaciones recientes, como [4], usaron Faster R-CNN para mejorar la detección de señales de tráfico en condiciones de baja resolución.

## 2.2. Modelos de clasificación de imágenes

Las CNN han sido el estándar en clasificación de imágenes gracias a su capacidad para aprender características jerárquicas. En el contexto del reconocimiento de señales de tráfico, los modelos preentrenados son una herramienta crucial:

- **ResNet[5]:** Su diseño de bloques residuales ayuda a evitar el problema de la degradación en redes profundas. ResNet-50 y ResNet-101 han sido ampliamente utilizados para clasificar señales del conjunto de datos GTSRB, alcanzando altas precisiones con ajustes finos (*fine-tuning*).
- **EfficientNet[14]:** Este modelo optimiza la relación entre precisión y eficiencia computacional, destacando en dispositivos con recursos limitados.

Los modelos basados en *Transformers*, como *Visual Transformers* (ViT)[3] han irrumpido como una alternativa a las CNN, mostrando un rendimiento sobresaliente en tareas de clasificación. En señales de tráfico, ViT ha demostrado ser robusto frente a condiciones adversas, como iluminación variable y señales desgasadas[16].

## 2.3. Técnicas de preprocessamiento de imágenes

El preprocessamiento de imágenes es crucial para mejorar la robustez de los modelos frente a variaciones ambientales

- **CLAHE** (*Contrast Limited Adaptive Histogram Equalization*): Esta técnica mejora la visibilidad de las señales bajo condiciones de iluminación adversas al aumentar el contraste local en las imágenes[15].
- **Data augmentation:** Métodos como rotación, escalado, y cambios de color ayudan a entrenar modelos más robustos frente a la variabilidad de las señales.
- **Normalización y filtrado:** Operaciones como la conversión a espacios de color *GRAY* son útiles para resaltar características relevantes, centrándose más en el contenido de la señal y no tanto en el color, mejorando la tarea de clasificación[9].

## 2.4. Implementaciones en Tiempo Real

El despliegue de modelos en vehículos autónomos requiere optimización para hardware específico. *TensorRT* y *ONNX Runtime* son herramientas que permiten comprimir y acelerar modelos para ejecutarlos en dispositivos como GPUs NVIDIA Jetson.

Por otro lado, también se ha desarrollado modelos ligeros como *YOLO-Tiny* y *MobileNet* que han sido adaptados para escenarios de recursos limitados, manteniendo un balance entre precisión y velocidad[7].

Este estado del arte demuestra que el campo del reconocimiento de señales de tráfico está en constante evolución, impulsado por los avances en modelos de aprendizaje profundo y las necesidades de sistemas inteligentes en tiempo real.

## 3. Metodología

Este proyecto consta de dos partes. En primer lugar se estudiarán diferentes modelos YOLO que sean capaces de detectar señales de tráfico en la carretera. Se buscarán modelos preentrenados y no se realizará el entrenamiento de un nuevo modelo debido a la falta de recursos y tiempo necesarios para ello.

Estos modelos serán estudiados con imágenes sacadas del *Street View* de Google Maps, ya que son imágenes parecidas a las que serán presentadas a nuestro modelo final por parte de la cámara instalada en nuestro vehículo. En este proceso, estudiaremos si los modelos detectan de verdad todas las señales de tráfico de las imágenes, independientemente de la clase que detecte el modelo, si es que detecta alguna clase. Esto se debe a que nuestro modelo CNN se encargará de clasificar estas señales en sus categorías, por lo que solo nos interesa que los modelos detecten bien todas las señales.

En segundo lugar, se utilizará un conjunto de datos de señales de tráfico para entrenar al modelo CNN que realizará la clasificación de estas en sus respectivas clases. Para ello se ha utilizado el dataset de *Kaggle GTSRB - German Traffic Sign Recognition Benchmark* [10], que incluye más de 50000 imágenes pertenecientes a 43 clases entre las que se encuentran diferentes límites de velocidad, ceda al paso, prohibido, cuidado resaltos, entre otras.

Para la clasificación de las imágenes se probará con modelos simples, algo más complejos o preentrenados como *ResNet*. También se introducirán y compararán técnicas de preprocessamiento como CLAHE para mejorar el rendimiento de estos. El rendimiento se evaluará con un conjunto de fotos de señales de tráfico españolas que, como es lógico, el modelo no ha visto previamente. Se elegirá el modelo que mejor clasifique estas imágenes.

Por último, una vez tenemos los dos modelos se grabará un vídeo que demostrará el funcionamiento en tiempo real de la aplicación. Este vídeo consistirá en un recorrido típico de un vehículo por las inmediaciones del Campus Sur de la Universidad Politécnica de Madrid.

## 4. Experimentos y Resultados

Empezaremos hablando de la elección de los modelos YOLO que serán capaces de detectar señales de tráfico. Como se ha comentado antes, se estableció que la detección de señales fuese realizada por un modelo YOLOv8 preentrenado para esta tarea. El primer objetivo fue que el modelo simplemente reconociese señales de tráfico además de otros posibles objetos, pero no fue posible encontrar un modelo así. Por tanto, se escogieron modelos que además de detectar señales, fuesen capaces de clasificarlas en tipos; sin embargo, independientemente del tipo, todas las detecciones se considerarían igual, como señales 'sin tipo'. Para evaluar los diferentes modelos encontrados se utilizaron imágenes del *Google Street View* de las inmediaciones del Campus Sur de la UPM (Figuras 1 y 2), lugar donde se probó el proyecto final.



Figura 1: Campus Sur 1



Figura 2: Campus Sur 2

El primer modelo [2] solo era capaz de detectar señales de velocidad, y aun así, no era demasiado preciso con las clases. De todas formas, esto no es algo importante, pues el objetivo primordial era que detectase correctamente el máximo número de señales posible, independientemente del tipo. El segundo modelo [13] fue capaz de detectar alguna señal más como prohibiciones o peligro, pero no el total de las señales. Finalmente con el tercer modelo [11] se consiguió detectar todas las señales de tráfico visibles en la imagen.



Figura 3: Señales de prueba

Por otro lado, para el desarrollo de los modelos CNN de clasificación de señales se optó por seguir una ruta de distintos modelos a cada cual más complejo. Previamente, como preprocessamiento de los datos[10], se estableció un tamaño común de 50x50, la media de los distintos tamaños de las imágenes, y se normalizaron los valores de los píxeles entre 0 y 1. Para su evaluación se han elegido una serie de imágenes de señales de tráfico españolas de *Google Maps Street View*, con las que se probará el rendimiento de los modelos en sus tareas de clasificación (Figura 3).

Para el primer modelo se optó por una arquitectura sencilla, de tres capas convolucionales seguidas cada una de un *MaxPooling 2x2* y dos capas densas con un *Dropout* entre ellas, la última con las 43 neuronas correspondientes a las 43 clases del conjunto de datos. La función de activación de la última capa fue *softmax* y las del resto *ReLU*. La función de loss establecida es *sparse\_categorical\_crossentropy* por el tipo de datos de entrada y de optimizer *adam*. Estos hiperparámetros se establecerán de la misma forma en los siguientes modelos. Se consigue un *accuracy* de 98 % sin *overfitting* (Figura 4).

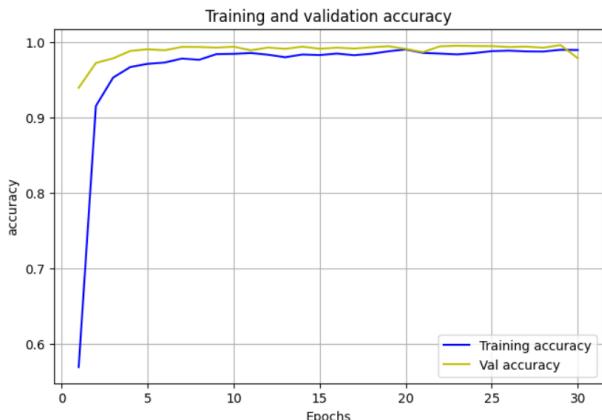


Figura 4: Accuracy Model 1

Pero probando el modelo con las imágenes de la Figura 3 se observa que no generaliza bien y no clasifica bien bastantes imágenes.

Para el segundo modelo se emplearon 4 capas convolucionales con *BatchNormalization* entre ellas y tres capas densas incluyendo la capa final de 43 neuronas. También se incluyen alguna capa de *MaxPooling* entre capas, así como de *Dropout*. También se pasan las etiquetas a *one-hot* y se utiliza la función de coste *categorical\_crossentropy*.

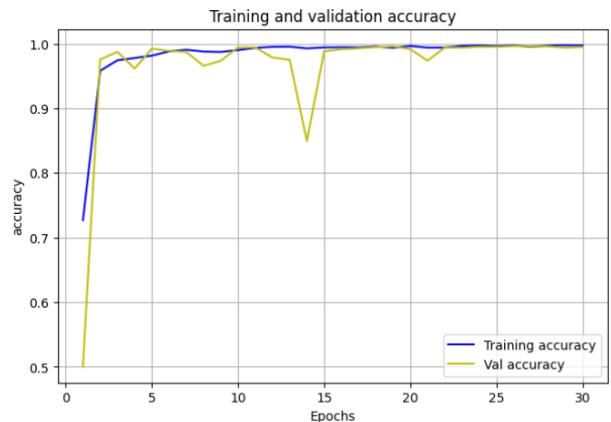


Figura 5: Accuracy Model 2

Se observa un *accuracy* muy bueno, del 98 % (Figura 5) aunque tanto el de entrenamiento como el de validación son buenos el de validación es ligeramente peor, lo que podría indicar *overfitting*. Observando su rendimiento en las imágenes de señales españolas nos damos cuenta que sigue fallando en bastantes, cometiendo errores que no se pueden permitir en una aplicación de este tipo.

Para mejorar estos resultados, se realiza un estudio más profundo de los datos, los cuales presentan dos problemas. Las imágenes tienen poca resolución, y además su contraste es pobre, por lo que aplicaremos

el método CLAHE para mejorar su contraste. Además, probaremos también a pasar las imágenes a blanco y negro para que se fije más en la forma de cada señal y no tanto en el color (Figura 6).

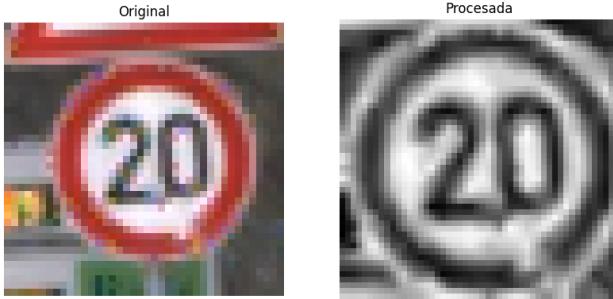


Figura 6: Imagen Original vs Procesada

Por otra parte nos dimos cuenta que el dataset está totalmente desbalanceado, algunas clases cuentan con alrededor de 2000 imágenes, mientras que otras apenas pasan de 200. Se procede a su balanceo utilizando la técnica de *Data Augmentation*. Este nuevo modelo con un mejor preprocesamiento cuenta con la misma arquitectura que el anterior.

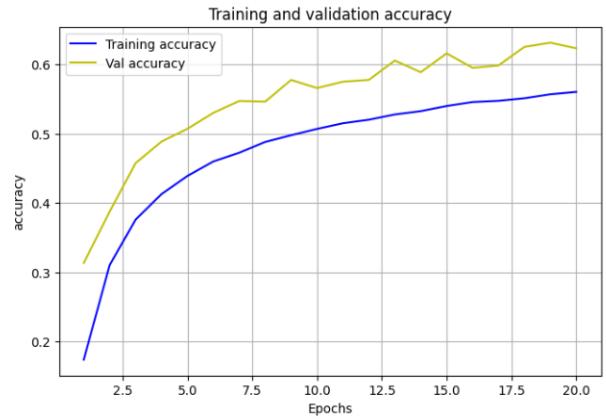


Figura 8: Accuracy Model 4

A pesar de que no hay *overfitting*, apenas se pasa del 60 % de *accuracy*, y como bien indicaban estos resultados, la predicción con las señales españolas no es buena, cometiendo más errores que aciertos.

Concluimos que el modelo con CLAHE y las imágenes en blanco y negro es el mejor, por lo que, haciendo uso también del modelo YOLO elegido, probamos estos con una imagen de la calle del campus, lugar donde se grabará el vídeo final (Figura 9).

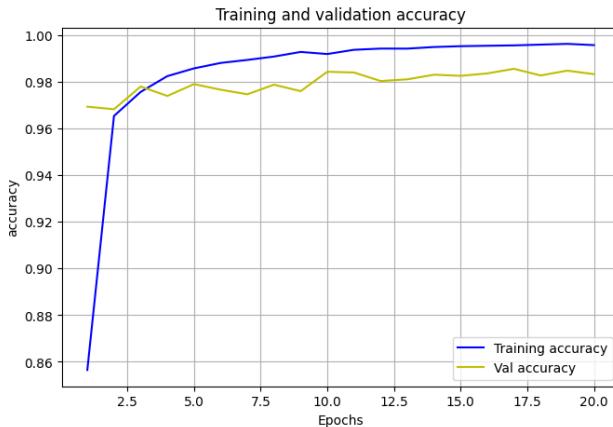


Figura 7: Accuracy Model 3

Sin embargo, a pesar de superar el 98 % (Figura 7) de *val\_accuracy*, el *training accuracy* le supera, dando la impresión de que hay *overfitting* de nuevo. No obstante, los resultados con las imágenes españolas son muy buenos, pues predice bien prácticamente todas las señales, salvo una. La señal mal predicha es debido a que los modelos han sido entrenados con imágenes de señales de tráfico alemanas y ahora los estamos probando con imágenes de calles y señales españolas, y concretamente, dicha señal es distinta en Alemania y España.

A pesar de los buenos resultados, intentamos mejorar la arquitectura haciendo uso de la técnica *Transfer Learning*, utilizando el modelo *ResNet50*. A este modelo le añadiremos un *GlobalAveragePooling* y dos capas densas incluyendo la última de clasificación. Los resultados son los peores obtenidos (Figura 8).

## 5. Conclusiones

El trabajo realizado demuestra que es viable utilizar modelos basados en aprendizaje profundo para la detección y clasificación de señales de tráfico, elementos esenciales en los sistemas de conducción autónoma y asistencia al conductor. A lo largo del estudio, se evaluaron diferentes versiones de YOLO, siendo uno basado en YOLOv8 el modelo seleccionado por su capacidad para detectar señales de tráfico en tiempo real. Aunque inicialmente hubo dificultades para hallar un modelo preentrenado que cubriera todas las necesidades, el modelo final fue capaz de identificar la mayoría de las señales presentes en las imágenes de prueba, cumpliendo con el objetivo.

En cuanto a la clasificación, se diseñaron varias arquitecturas de redes neuronales convolucionales que



Figura 9: Ejemplo detección YOLO + CNN

mostraron altos niveles de precisión en entornos controlados. Sin embargo, las pruebas con imágenes de señales españolas no vistas durante el entrenamiento evidenciaron problemas de generalización. Para mejorar estos resultados, se implementaron técnicas de preprocessamiento como CLAHE, que optimizó el contraste de las imágenes, y se aplicó *Data Augmentation* para balancear el dataset. Estas estrategias permitieron que el modelo mejorara significativamente su rendimiento, alcanzando altos niveles de precisión y logrando clasificar correctamente casi todas las señales españolas en condiciones reales.

El uso de *Transfer Learning* con *ResNet50* ofreció resultados inferiores en este caso particular. A pesar de que la técnica permitió una mejor generalización, el modelo no alcanzó la precisión esperada con las señales españolas, resaltando la importancia de contar con datasets más diversos que reflejen las diferencias culturales entre países.

La combinación del modelo YOLOv8 para la detección y una CNN con técnicas de preprocessamiento para la clasificación resultó ser funcional en escenarios reales, como se demostró al hacer la prueba con un vídeo grabado en las inmediaciones del Campus Sur de la Universidad Politécnica de Madrid. Aunque se identificaron limitaciones específicas, como las dificultades para clasificar señales únicas de un país, estos problemas pueden ser abordados en futuros trabajos mediante ajustes en el dataset y mejoras en los modelos utilizados.

Para finalizar, este proyecto representa un avance importante hacia la implementación de sistemas autónomos robustos para la detección y clasificación de señales de tráfico en tiempo real, destacando la necesidad de seguir investigando en áreas como la diversidad del dataset y la optimización de los modelos para escenarios aún más desafiantes. Los resultados obtenidos son prometedores y sientan una base sólida para futuros desarrollos en el ámbito de la conducción autónoma y los sistemas avanzados de asistencia al conductor.

## Referencias

- [1] Alexey Bochkovskiy, Chien-Yao Wang y Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: <https://arxiv.org/abs/2004.10934>.
- [2] Parisa Karimi Darabi. *Traffic Signs Detection Using YOLOv8*. URL: <https://www.kaggle.com/code/pkdarabi/traffic-signs-detection-using-yolov8>.
- [3] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: <https://arxiv.org/abs/2010.11929>.
- [4] Xiang Gao et al. "Improved Traffic Sign Detection Algorithm Based on Faster R-CNN". En: *Applied Sciences* 12.18 (2022). ISSN: 2076-3417. DOI: 10.3390/app12188948. URL: <https://www.mdpi.com/2076-3417/12/18/8948>.
- [5] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: <https://arxiv.org/abs/1512.03385>.
- [6] Bin Ji et al. "Improved YOLOv8 for small traffic sign detection under complex environmental conditions". En: *Franklin Open* 8 (2024), pág. 100167. ISSN: 2773-1863. DOI: <https://doi.org/10.1016/j.fraope.2024.100167>. URL: <https://www.sciencedirect.com/science/article/pii/S2773186324000975>.
- [7] Zicong Jiang et al. *Real-time object detection method based on improved YOLOv4-tiny*. 2020. arXiv: 2011.04244 [cs.CV]. URL: <https://arxiv.org/abs/2011.04244>.
- [8] Rahima Khanam y Muhammad Hussain. *What is YOLOv5: A deep look into the internal features of the popular object detector*. 2024. arXiv: 2407.20892 [cs.CV]. URL: <https://arxiv.org/abs/2407.20892>.
- [9] Barney Kim. *Traffic Sign Recognition*. URL: <https://medium.com/@wolfapple/traffic-sign-recognition-2b0c3835e104>.
- [10] Mykola. *GTSRB - German Traffic Sign Recognition Benchmark*. URL: <https://www.kaggle.com/datasets/meowmeowmeowmeow/gtsrb-german-traffic-sign/data>.
- [11] Duy Pham. *Traffic Detection With YOLOv8*. URL: <https://www.kaggle.com/code/duy18102004/traffic-detection-with-yolov8>.
- [12] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: <https://arxiv.org/abs/1506.01497>.
- [13] Phillip Ssempeeewa. *YOLOv8 Road Sign Detection*. URL: <https://www.kaggle.com/models/phillipssempeeewa/yolov8-road-sign-detection?select=best.pt>.
- [14] Mingxing Tan y Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG]. URL: <https://arxiv.org/abs/1905.11946>.
- [15] Thomas Tracey. *Recognizing Traffic Signs with CNNs*. URL: <https://medium.com/@thomastracey/recognizing-traffic-signs-with-cnns-23a4ac66f7a7>.

- [16] Haolan Wang. "Traffic Sign Recognition with Vision Transformers". En: *Proceedings of the 6th International Conference on Information System and Data Mining*. ICISDM '22. Silicon Valley, CA, USA: Association for Computing Machinery, 2022, págs. 55-61. ISBN: 9781450396257. DOI: 10.1145/3546157.3546166. URL: <https://doi.org/10.1145/3546157.3546166>.
- [17] Jiaao Xia et al. "DSRA-DETR: An Improved DETR for Multiscale Traffic Sign Detection". En: *Sustainability* 15.14 (2023). ISSN: 2071-1050. DOI: 10.3390/su151410862. URL: <https://www.mdpi.com/2071-1050/15/14/10862>.
- [18] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV]. URL: <https://arxiv.org/abs/2408.15857>.