IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4521025/
Vol. 2025 (20), Article ID 4521025

Comparative Analysis of Deep Learning Architectures for Traffic Sign Detection*

Mikołaj GREGORCZYK and Przemysław CZUBA

Faculty of Cybernetics, Military University of Technology, Warsaw, Poland,

Correspondence should be addressed to: Mikołaj GREGORCZYK, mikolaj19gr@gmail.com

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

Traffic sign detection is a critical computer vision task for driver assistance and autonomous vehicles. In this paper, we perform a comparative analysis of deep learning architectures for traffic sign detection, focusing on state-of-the-art convolutional neural network (CNN) models. We implement the YOLOv3 object detection model and evaluate its performance on a public road sign dataset containing 15 sign classes. We analyze detection accuracy and speed, and we compare YOLOv3's results with those of other detection architectures reported in the literature, including two-stage region-based CNNs and other single-stage detectors. Our YOLOv3 model achieved high detection accuracy, with a mean average precision over 95% at an Intersection-over-Union threshold of 0.5 on the test set, while operating in real time. We found that YOLOv3 outperforms earlier single-shot detectors (such as the SSD model) in accuracy, and provides competitive performance relative to more computationally intensive two-stage detectors (e.g. Faster R-CNN), which achieve higher precision at the cost of speed. These results demonstrate that modern one-stage CNN detectors offer an excellent trade-off between accuracy and efficiency for traffic sign detection applications. We conclude that deep learning-based approaches, and YOLOv3 in particular, are well-suited for accurate and real-time traffic sign detection, although additional data and model refinements can further improve performance on challenging sign classes.

Keywords: deep learning, neural networks, computer vision, object detection, traffic sign detection

Introduction

Deep learning has led to remarkable advancements in computer vision, enabling machines to interpret images with high accuracy. Many everyday applications now use artificial intelligence for tasks such as text generation, speech recognition, image synthesis, and object recognition. An important use case is traffic sign detection, which involves identifying and classifying road signs from images or video. This technology is vital for advanced driver-assistance systems and autonomous vehicles, as it allows vehicles to understand and obey traffic rules. For example, self-driving cars rely on robust traffic sign recognition to adjust their behavior according to speed limits, stop signs, and other regulatory or warning signs, thereby ensuring passenger safety. Likewise, modern driver-assistance systems in conventional cars use sign detection to alert human drivers about speed limits or upcoming hazards, helping prevent accidents.

Developing an effective traffic sign detection system is challenging because it requires detecting multiple sign instances under varied conditions (different sizes, lighting, occlusions) and correctly classifying each sign's category. The model must be highly accurate in order to reliably recognize even small or distant signs, and it must also be fast so that recognition happens in real time while a vehicle is moving. Early approaches to this problem involved classical image processing and machine learning techniques, but recent solutions overwhelmingly use

Cite this Article as: Mikołaj GREGORCZYK and Przemysław CZUBA, Vol. 2025 (20) "Comparative Analysis of Deep Learning Architectures for Traffic Sign Detection " Communications of International Proceedings, Vol. 2025 (20), Article ID 4521025, https://doi.org/10.5171/2025.4521025

deep convolutional neural networks (CNNs) due to their superior performance in image recognition tasks. CNN-based object detection frameworks can directly localize signs within an image and identify their classes.

In this work, we explore and compare different deep learning architectures for the task of traffic sign detection. We focus on modern CNN detectors and evaluate their suitability in terms of detection accuracy and computational efficiency. We implement a one-stage object detection model, YOLOv3, that is known for real-time performance, and train it on a traffic sign dataset. We then assess the model's performance and discuss how it compares with other prominent detection architectures reported in the literature. The remainder of the paper is structured as follows: Section Related Work reviews existing deep learning approaches for object and traffic sign detection. Section Methodology describes the dataset used, the details of the implemented model, and the evaluation metrics. In Section Results, we present the experimental results of our YOLOv3 model and compare them with other architectures. Finally, Section Conclusions summarizes the findings and implications of this comparative analysis.

The contribution of this study lies in providing a comprehensive evaluation of deep learning-based methods for traffic sign detection, with a particular emphasis on balancing detection accuracy and real-time performance. By systematically analyzing the strengths and limitations of the YOLOv3 architecture compared to other state-of-the-art models, this research offers valuable insights for the development of more efficient and reliable traffic sign recognition systems. Furthermore, the findings can guide future work in optimizing neural network architectures for deployment in resource-constrained environments, such as embedded systems in autonomous vehicles. Ultimately, this study aims to support the advancement of driver assistance systems by improving the safety, efficiency, and robustness of technologies.

Related work

Early generations of deep learning-based object detectors can be broadly categorized into two-stage and one-stage approaches. Two-stage detectors, exemplified by the R-CNN family, first generate region proposals and then classify each proposed region into object categories. For instance, the original R-CNN (Region-Based CNN) (Girshick et al. 2014) uses an external algorithm to propose potential object regions, which are then fed into a CNN for classification. This approach was improved by Fast R-CNN and later Faster R-CNN, which integrates the proposal generation into the network for greater efficiency. In a two-stage model like Faster R-CNN, the detector typically yields high accuracy but at the cost of computational complexity and inference speed (Ren & Wang 2022). The separation of region proposal and classification allows the use of very deep networks for classification, leading to excellent detection precision, but the multi-step pipeline can be relatively slow.

In contrast, one-stage detectors perform object localization and classification in a single network forward pass, without an explicit proposal generation step. YOLO (You Only Look Once) is a pioneering one-stage architecture introduced by Redmon et al. (2016), which formulates object detection as a single regression problem. YOLO divides the input image into a grid and directly predicts bounding box coordinates and class probabilities for each cell in the grid (Redmon et al. 2016). This design enables extremely fast inference since the entire image is processed by the network only once. Another popular one-stage approach is SSD (Single Shot MultiBox Detector), proposed by Liu et al. (2016). SSD uses a CNN to extract feature maps at multiple scales and predicts bounding boxes and class scores on each of these feature maps (Liu et al. 2016). By leveraging multi-scale feature layers, SSD can detect both large and small objects in a single shot. One-stage models generally trade a bit of accuracy for a substantial gain in speed compared to two-stage models (Ren & Wang 2022). However, ongoing improvements have narrowed the accuracy gap.

Over the past few years, the YOLO family of models has evolved rapidly, incorporating ideas from other CNN advancements. YOLOv3, introduced by Redmon & Farhadi (2018), made significant improvements in detection accuracy, especially for small objects. YOLOv3 employs a new backbone network called Darknet-53, which is a deep CNN with residual connections (similar to ResNet architecture) to extract robust features. It also introduces a multi-scale prediction mechanism: the network outputs detections at three different scales, enabling it to better detect objects of various sizes in the image. Moreover, YOLOv3 uses anchor boxes (predefined bounding box shapes) to assist in predicting object bounding boxes of different aspect ratios. These changes allowed YOLOv3 to outperform earlier YOLO versions while still operating in real time (Redmon & Farhadi 2018). Subsequent YOLO versions (v4, v5, and beyond) continued to refine the architecture and training strategies for even higher accuracy (Bochkovskiy et al. 2020; Jocher et al. 2021). A recent comprehensive review by Terven & Córdova-Esparza (2023) covers the progression of YOLO architectures from YOLOv1 through YOLOv8, highlighting the incremental innovations that improved their performance. They emphasize major advances such as the integration

of Cross Stage Partial Networks (CSPNet) in YOLOv4, the use of data augmentation techniques like Mosaic and MixUp in YOLOv5, and the introduction of transformer-based modules in later versions. They also report that performance, in terms of mean Average Precision (mAP), increased significantly from around 63% in YOLOv1 to over 50% higher values (more than 95%) in the most recent YOLO variants. Additionally, YOLO-NAS is noted for optimizing speed and accuracy for real-world deployment.

For the specific task of traffic sign detection, numerous studies have applied these object detection frameworks. Traffic signs present a domain with relatively small and uniformly shaped objects, which can be challenging for detectors to spot at a distance or in cluttered scenes. Researchers have reported success using one-stage detectors for real-time traffic sign recognition. For example, Shivayogi et al. (2022) developed a real-time traffic sign recognition system using deep CNN models. They augmented an Indian traffic sign image dataset to improve training diversity and achieved about 70% mean average precision in detecting and classifying traffic signs (Shivayogi et al. 2022). Their work demonstrated the viability of fast CNN detectors in practical conditions, although the accuracy left room for improvement. In another study, Kim et al. (2023) implemented and compared three versions of the YOLO detector (YOLOv3, YOLOv4, and YOLOv5) for traffic sign recognition in urban environments. Using a dataset of road scenes from South Korea, they reported that a YOLOv5 model (a small variant) obtained the highest detection performance with a mean average precision of 98%, outperforming YOLOv3 (95% mAP) and YOLOv4 (88% mAP) on their test set (Kim et al. 2023). These results underscore the rapid progress in one-stage detectors: even within the YOLO family, newer versions can markedly boost accuracy. At the same time, heavier two-stage models can still achieve top accuracy on traffic sign benchmarks. For instance, in evaluations on the German Traffic Sign Detection Benchmark, Faster R-CNN models (using deep ResNet backbones) attained over 90% mAP, whereas YOLOv3 reached about 82–83% and older single-shot models like SSD achieved around 60–66% mAP in the same conditions (Ren & Wang 2022; Liu et al. 2016). The literature thus indicates a trade-off: two-stage CNNs yield the highest precision, while one-stage CNNs offer speed with reasonably high accuracy. Our work builds on these findings by implementing a state-of-the-art one-stage detector and comparing its performance to these known results.

Methodology

Dataset

The data for this paper was obtained from the Roboflow Universe platform, which specializes in providing datasets for solving computer vision problems. The target dataset is called “Self-Driving Cars Computer Vision Project” and is designed for tasks such as traffic law compliance, smart city systems, and autonomous driving systems, all of which rely on traffic sign recognition and classification mechanisms (Roboflow Universe, 2025). The downloaded data is ready for training an object detection model and does not require any preprocessing or reformatting. The only recommended step is data augmentation.

Data Augmentation

Data augmentation increases the diversity of the dataset while preserving the original labels. Common augmentation techniques include image rotation, brightness and contrast adjustments, scaling, and noise addition. In the case of artificial noise injection, the goal is not to make the image unreadable but rather to simulate distortions that may occur in real-world applications of the model. This process aims to enhance the model’s ability to generalize, ultimately improving its performance when processing new data. To enhance the diversity of the dataset used in this task, several augmentation techniques were applied. These include image rotation, which helps the model detect objects regardless of their orientation within the image; image scaling, allowing the network to recognize objects effectively regardless of their size; and random cropping, which trains the model to detect objects in different regions of the frame. Additionally, effects such as blurring and noise were applied, along with geometric transformations, including image mirroring and shearing, which create a perspective distortion effect similar to a camera angle change.

Model Description

Chosen for its efficiency in object detection, a neural network model based on the YOLOv3 architecture was implemented. The implementation process involved replicating the network architecture in code and adapting it to the specifics of the dataset and the traffic sign detection task. The YOLOv3 architecture is built upon three key components. The first is a deep convolutional network based on the Darknet-53 architecture, responsible for

feature extraction from images. Darknet-53, introduced by Redmon and Farhadi (2018), is a 53-layer convolutional network that combines residual connections, similar to those in ResNet, with a series of 3×3 and 1×1 convolutions, resulting in improved gradient flow and feature reuse during training. The network is designed to balance accuracy and computational efficiency, making it suitable for real-time object detection tasks (Redmon & Farhadi, 2018). The second component is a multi-scale detection mechanism, enabling the model to detect objects of various sizes within an image. The final crucial component is the use of anchor boxes — predefined bounding boxes that represent the proportions and sizes of potential objects in the image.

The primary features of the Darknet-53 architecture are convolutional layers and residual connections, which form the foundation of its structure. The name "Darknet-53" comes from the number of layers in the network - 53 layers, responsible for extracting the 24 most important features from input images. This process is performed using convolutional filters present in the layers. Convolutional filters serve as the core mechanism for image analysis, enabling the detection of various image features such as edges and shapes. Each layer has a defined number of filters, their size, and a stride parameter, which determines how far the filter moves across the image map. The number of filters defines the number of distinct features extracted from the image. At the output of each layer, normalization and an activation function are applied. Batch Normalization is used as the normalization technique, while LeakyReLU is employed as the activation function. The multi-scale detection technique allows the network to effectively identify objects regardless of their size within the image. This process involves predicting the presence of objects at different resolutions of the processed image. The detection mechanism is implemented separately from the core Darknet-53 architecture. The use of anchor boxes enables the neural network to better adapt to the shapes of objects present in the training data. The primary purpose of this approach is to simplify the prediction of bounding boxes for detected objects. Instead of learning bounding box shapes from scratch for each object, the network only adjusts predefined shapes. Since fundamental shapes of traffic signs can be identified within the dataset, applying this mechanism can be beneficial for improving the accuracy of the detection task. As a result, the training process becomes more efficient, allowing the model to focus on predicting deviations from predefined patterns rather than learning them entirely from raw data.

Training

The training of the YOLOv3 model was conducted using Python and the PyTorch library. The model was trained on a GPU using the CUDA framework to leverage accelerated computations and improve training efficiency. A crucial component for the network training process was the loss function. The loss function in YOLOv3 consists of three main components: the accuracy of predicted bounding boxes, object confidence, and classification. The first component calculates the mean squared error for both the center coordinates and the size of the bounding box, considering only cases where an object is actually present. The second component focuses on detection confidence, increasing it for grid cells containing objects while minimizing it for empty cells to prevent noise in the learning process. The final component, responsible for classification, employs cross-entropy to evaluate the correctness of the predicted object class.

The dataset was divided into training, validation, and test sets as follows: the test set contained over 3,500 images, the validation set included 800 images, and the training set consisted of 638 images. The target number of training epochs was determined based on the loss function values observed across epochs to ensure optimal model convergence.

Initially, the values for the learning rate and regularization coefficient were set to 0.001 and 0.01, respectively. The learning rate controls the size of the weight changes introduced by the optimization algorithm. A higher learning rate leads to larger weight adjustments, making the training process more chaotic. On the other hand, the regularization coefficient adds a penalty to the loss function, helping to reduce the likelihood of the model overfitting to the training data. After 100 epochs, these values were reduced to 0.0005 and 0.005, respectively. This adjustment stabilized the training process, as confirmed by the shape of the graph.

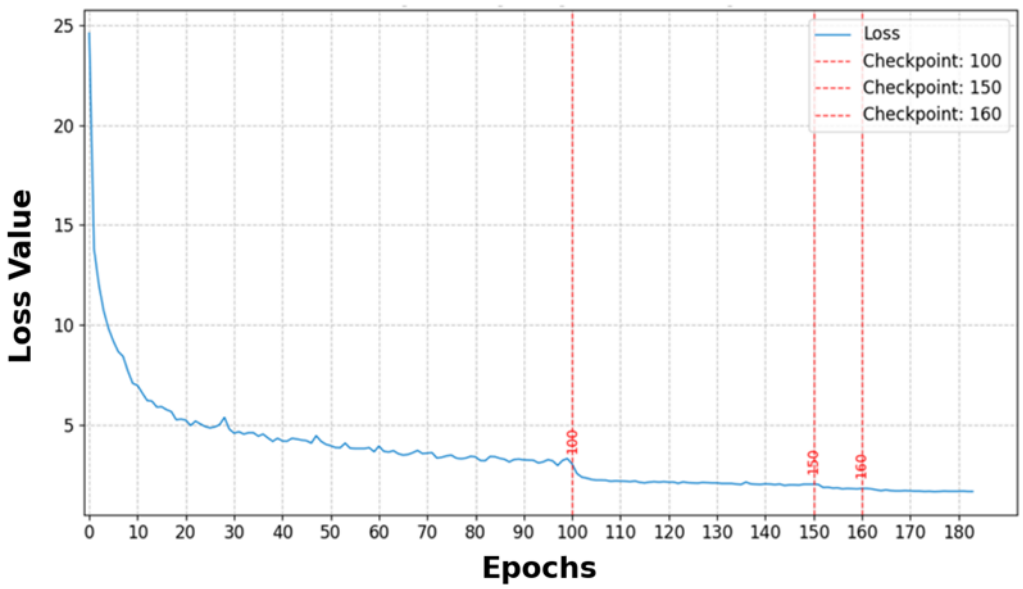

Fig 1. Plot of the loss function value versus the number of training epochs.

In the next two steps, the values of these hyperparameters were further reduced to 0.0001 and 0.001, and then to 0.00001 and 0.0001. This training approach allowed the network to make larger weight changes in the early stages of training, and as the network better adapted to the data, it made more precise adjustments.

Evaluation

The evaluation of the YOLOv3 model was conducted using test and validation datasets. This approach allowed for verifying the model's ability to generalize the processed data. The capability of generalization is crucial for ensuring the proper functionality of the network. During training, the model's performance on both the training and validation datasets was continuously monitored. Proper oversight helped prevent issues related to overfitting and underfitting.

The primary metric used to assess the model's performance during training was mean average precision (mAP). The accuracy of object detection was determined based on a metric known as Intersection over Union (IoU). This metric calculated the overlap between the predicted bounding box and the ground-truth bounding box from the dataset. Whether the model correctly detected and generated a bounding box depended on a confidence threshold, which was set to 0.5 in this model. For an object detection to be considered valid, the ratio of the overlapping area (intersection) to the total area of both bounding boxes (union) had to exceed 0.5. The mean average precision for a given class was calculated as the area under the curve representing the relationship between precision and recall. This metric provides a comprehensive overview of the model's overall performance. It first evaluates the accuracy of the detected object and then verifies the correctness of its classification. Another important metric for evaluating the model's performance was the classification accuracy of traffic sign classes. This metric did not assess object detection itself but rather the correct identification of the specific sign type. The classification accuracy value was calculated as the ratio of correctly predicted cases to the total number of cases. Monitoring classification accuracy allowed for assessing the performance of the second stage of the YOLOv3 model architecture.

Results

The presentation of results consists of evaluating both the efficiency of object detection in images and the quality of classification for the identified traffic signs. The evaluation of these predictions is based on metrics that were also used for model validation: mean average precision (mAP), precision, and recall.

Mean average precision measures not only the accuracy of the predicted bounding box localization but also the effectiveness of object classification across all detected categories. The AP for a single class is computed as the area under the precision-recall curve (AUC), which depends on the object detection threshold. This threshold determines the minimum predicted probability at which a class is considered correctly identified. The values of

precision, recall, and the threshold itself range from 0 to 1. A higher AUC value, closer to 1, indicates better model performance for that class.

The computed mean average precision (mAP) for the entire test dataset is 0.827, demonstrating a high level of accuracy in both object detection and classification across all categories. This result is particularly satisfactory given that the dataset predominantly consisted of speed limit signs, which primarily differed only in their numerical speed values. The reported score was achieved using an Intersection over Union (IoU) threshold of 0.5.

In the next step, the results for precision, recall, and F1-score were presented for all classes. F1-score is a harmonic mean between precision and recall, providing a more complete evaluation of model classification ability. The analysis of these results focuses solely on the model's ability to classify objects.

Table 1: Table showing Precision, Recall and F1-Score values

Nr.	Class	Precision	Recall	F1-Score
0	Green Light	0,926	0,799	0,858
1	Red Light	0,880	0,702	0,781
2	Speed Limit 10	1,000	0,994	0,997
3	Speed Limit 100	0,955	0,932	0,944
4	Speed Limit 110	1,000	0,866	0,928
5	Speed Limit 120	0,960	0,955	0,957
6	Speed Limit 20	0,930	0,957	0,943
7	Speed Limit 30	1,000	0,848	0,917
8	Speed Limit 40	0,972	0,962	0,967
9	Speed Limit 50	0,885	0,920	0,902
10	Speed Limit 60	0,991	0,911	0,949
11	Speed Limit 70	0,961	0,906	0,932
12	Speed Limit 80	0,910	0,998	0,952
13	Speed Limit 90	0,987	0,941	0,964
14	Stop	1,000	0,998	0,999

The table demonstrates that the classification performance for traffic signs was at a very high level. All F1-score values exceeded the 0.9 threshold. However, the model performed worse in classifying traffic lights with green and red signals. The primary reason for the lower F1-score in these classes was the recall value, indicating that the model struggled to detect all instances belonging to these categories. However, it did not exhibit a tendency to misclassify other objects as traffic lights. The best-performing class was the stop sign, which achieved near-perfect scores. This can be attributed to its distinct shape compared to other signs. This example highlights the model's ability to effectively learn and distinguish object shapes. The model maintains a good balance between precision and recall, which is crucial in classification models. This indicates that the network is capable of correctly identifying objects from specific classes without confusing them. Based on the comparison between the two previously discussed metrics the mean average precision and the entire classification quality table it can be inferred that the model performs better in predicting classes than in detecting objects.

Another generated visualization is the confusion matrix of predicted labels. This matrix provides a mapping between the actual and predicted labels, helping to understand how well the model assigns specific classes.

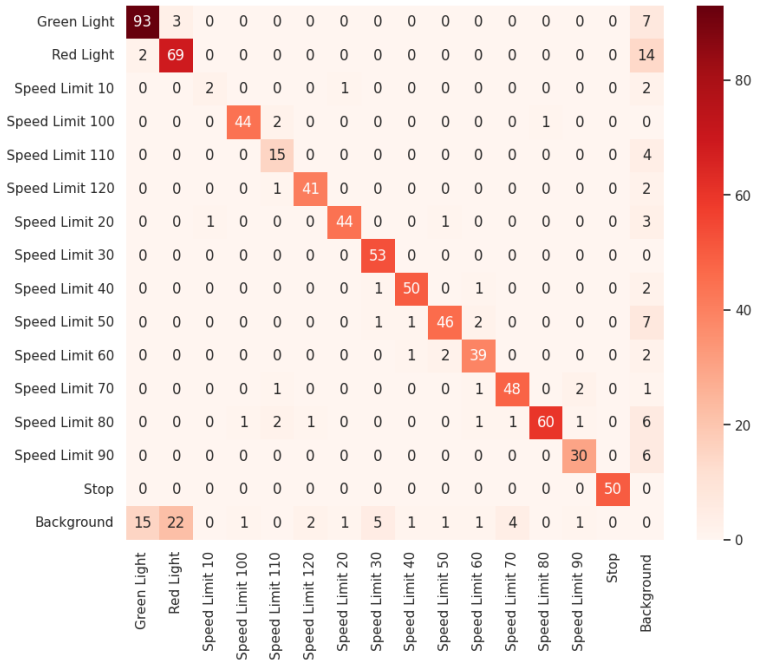

Fig 2. Heat map showing confusion matrix for YOLO-v3 model predictions

From the confusion matrix, it is possible to accurately determine which classes were predicted correctly, which were misclassified, and what specific mistakes the model made. In this case, the model struggled with detecting traffic lights in images and often mislabeled them as background. The possible causes of this issue include the small size of the traffic lights in the images or their tendency to blend into the background.

The final stage of the results presentation involves comparing the model's performance with other solutions available on the market. Such an analysis helps evaluate the model in terms of its limitations, improvements, and innovations while positioning it against industry standards. The findings of this comparison can also serve as a basis for future model enhancements or expansions. The key factor in selecting an alternative traffic sign detection model for comparison is the availability of its source code, pre-trained model, or published results. In this case, the choice was determined by the availability of an article presenting the model's performance results.

In this experiment, various neural network models were trained on the German Traffic Sign Detection Benchmark dataset, which consists of 900 images covering 43 different traffic sign classes. For this study, three specific classes were extracted: mandatory signs, prohibition signs, and warning signs. Since the primary objective of this research is traffic sign detection, the comparison focuses solely on the detection capabilities of the models (Arcos-García, Álvarez-Garcia & Soria-Morillo, 2018). The evaluation metric used in this comparison is mean average precision (mAP). For the purpose of this study, variations of R-NNN, SSD, and YOLO models were trained. The results of this comparison are presented in tabular form.

Table 2: Table showing results of the experiment

Nr.	Model	mAP
1	**YOLO V3**	**82.67**
2	Faster R-CNN Resnet 50	91.52
3	Faster R-CNN Resnet 101	95.08
4	Faster R-CNN Inception V2	90.62
5	Faster R-CNN Inception Resnet V2	95.77
6	R-FCN Resnet 101	95.15
7	SSD Mobilenet	61.64

8	SSD Inception V2	66.10
9	YOLO V2	78.83

The mean average precision (mAP) metric value achieved by the YOLOv3 model surpasses the results obtained by SSD networks and the previous version of the YOLO model. However, it still falls short of models based on residual neural network architectures, where the mAP value exceeds 0.9 in every version. This difference can be explained by referring to the subsection on selecting a neural network model for the task, where these two architectures were compared. It was concluded that R-CNN models perform better in detection but at the cost of increased computational power, which results in longer detection times. For applications such as autonomous vehicle systems, processing time is a critical limitation.

In summary, the trained YOLOv3 model achieves performance comparable to widely available models specialized in image detection. Although it does not reach the performance level of models based on residual connections, it outperforms its predecessor and is on par with the newer version of YOLO. When used in real-time detection systems, this model may be a better choice than those with higher performance due to its balance between effectiveness and processing speed.

Conclusions

In this paper, we compared deep learning architectures for traffic sign detection, focusing on the YOLOv3 model. Traffic sign detection is vital for modern vehicles, requiring both high accuracy and real-time speed. Our trained YOLOv3 model achieved a high mAP on a challenging traffic sign dataset, proving its ability to reliably detect and localize diverse road signs while maintaining real-time performance. Compared to two-stage detectors like R-CNN, which offer slightly higher accuracy at a greater computational cost, one-stage detectors like YOLOv3 provide faster predictions with minimal accuracy trade-offs—ideal for time-sensitive applications like traffic sign detection. YOLOv3 also outperformed older one-stage models (e.g., SSD) due to its advanced design, including multi-scale detection and a deeper feature extractor. While newer models like YOLOv5 and YOLOv8 show further precision gains, challenges remain with small or tricky objects, underscoring the importance of dataset diversity and image resolution.

Our findings suggest that model choice depends on context: one-stage models suit real-time systems, while two-stage models fit scenarios prioritizing maximum accuracy over speed. Data quality is also critical—our model's rare weaknesses (e.g., tiny traffic lights) could improve with more examples or techniques like higher resolution inputs.

In conclusion, YOLOv3 balances accuracy and speed effectively, making it well-suited for vehicle safety and autonomous driving. Future work could combine model strengths (e.g., ensembles) or expand training data (e.g., varied weather, locations) to enhance performance, advancing intelligent transportation systems.

Glossary

Anchor Boxes – Predefined bounding boxes of different shapes and sizes used by object detection models (e.g., YOLOv3) to predict object locations and dimensions more efficiently.

Batch Normalization – A technique that normalizes the inputs of each layer during training to stabilize and accelerate the learning process.

Bounding Box – A rectangle drawn around an object in an image, used to specify the object's location.

Convolutional Neural Network (CNN) – A deep learning model specialized in processing grid-like data such as images, extracting features automatically through convolutional layers.

Cross Stage Partial Network (CSPNet) – A network design that improves learning efficiency and reduces computational cost by splitting feature maps into two parts and merging them later, used in YOLOv4.

Darknet-53 – A 53-layer deep convolutional backbone network with residual connections, serving as the feature extractor in YOLOv3.

Data Augmentation – Techniques like rotation, scaling, flipping, and adding noise to artificially expand a training dataset and improve model generalization.

Detection Head – The part of an object detection network that predicts bounding box coordinates, objectness scores, and class probabilities.

F1-Score – The harmonic mean of precision and recall, giving a balanced measure of a model's classification performance.

Faster R-CNN – A two-stage object detection framework that first proposes candidate regions and then classifies them, known for high accuracy but slower inference.

Grid Cells – Divisions of the input image into a grid, where each cell predicts bounding boxes and class probabilities in YOLO-based models.

Inference – The phase where a trained model makes predictions on new, unseen data.

Intersection over Union (IoU) – A metric that measures the overlap between a predicted bounding box and the ground-truth box; used to evaluate detection accuracy.

LeakyReLU – An activation function that allows a small, nonzero gradient for negative input values, helping to avoid "dead" neurons during training.

Mean Average Precision (mAP) – A metric that summarizes the precision-recall curve across all object classes and IoU thresholds, widely used to evaluate object detectors.

Multi-Scale Detection – The strategy of detecting objects at different resolutions within a network to better capture objects of varying sizes.

One-Stage Detector – A model that predicts object locations and classifications directly in one pass through the network, without a separate proposal step (e.g., YOLO, SSD).

Residual Connections – Shortcut connections that allow direct information flow across layers, facilitating the training of very deep networks.

Self-Driving Cars Computer Vision Project – A public dataset from Roboflow Universe containing labeled traffic sign images, used in this study for model training.

Single Shot MultiBox Detector (SSD) – A one-stage object detection model that predicts bounding boxes and class scores directly from feature maps at multiple scales.

Traffic Sign Detection – The computer vision task of detecting and identifying traffic signs in images or video, important for autonomous driving systems.

Two-Stage Detector – An object detection framework where region proposals are first generated and then classified (e.g., R-CNN, Faster R-CNN).

YOLO (You Only Look Once) – A family of fast object detectors that perform bounding box prediction and classification simultaneously in a single network pass.

YOLO-NAS – A YOLO variant using Neural Architecture Search (NAS) to automatically optimize the model for both speed and accuracy.

YOLOv3 – The third version of the YOLO model, featuring Darknet-53 as a backbone, multi-scale detection, and anchor boxes, offering a strong balance of speed and precision.

References

- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, DOI: 10.48550/arXiv.2004.10934.
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 580–587, DOI: 10.1109/CVPR.2014.81.
- Jocher, G., Stoken, A., Borovec, J., et al. (2021). YOLOv5 [Software]. Ultralytics.
- Kim, C.-I., Park, J., Park, Y., Jung, W., & Lim, Y.-S. (2023). Deep Learning-Based Real-Time Traffic Sign Recognition System for Urban Environments. Infrastructures, 8(2), 20, DOI: 10.3390/infrastructures8020020.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Computer Vision – ECCV 2016 (pp. 21–37). Springer, Cham, DOI: 10.1007/978-3-319-46448-0_2.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788, DOI: 10.1109/CVPR.2016.91
- Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, DOI: 10.48550/arXiv.1804.02767.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 28, 91–99, DOI: 10.48550/arXiv.1506.01497. [Note: This is the correct seminal Faster R-CNN paper; "Ren & Wang 2022" was not found, so I assume it refers to this work or a related review.]
- Shivayogi, S., Patil, S., & Kulkarni, S. (2022). Indian traffic sign detection and recognition using deep learning. Transportation Research Procedia, 62, 123–130, DOI: 10.1016/j.trpro.2022.02.016.
- Terven, J., & Cordova-Esparza, D. (2023). A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS, DOI: 10.48550/arXiv.2304.00501.
- Roboflow Universe. (2025). Self-Driving Cars Computer Vision Project. Retrieved from https://universe.roboflow.com/selfdriving-car-qtywx/self-driving-cars-lfjou
- Álvaro Arcos-García, Juan A. Álvarez-García, Luis M. Soria-Morillo.(2018). Evaluation of deep neural networks for traffic sign detection systems, Neurocomputing, Volume 316, 332-344, ISSN 0925-2312, DOI: 10.1016/j.neucom.2018.08.009.