Improving Safety in Autonomous Driving with the Use of AI for Object Detection Prediction

Shahbaz Satti

Abstract

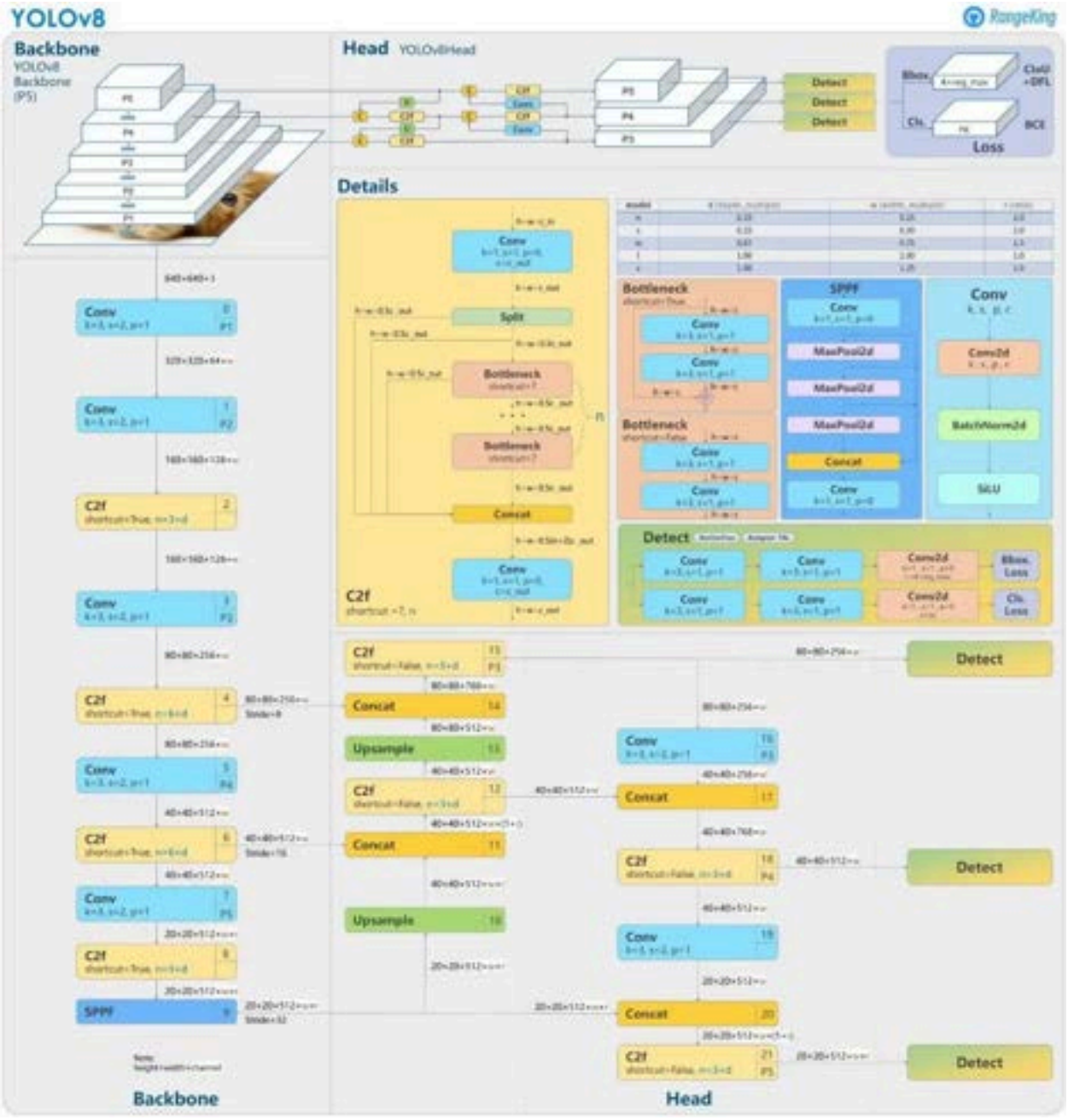
In the last decade, object detection and machine learning-based algorithms have enhanced significantly, making self-driving vehicles a reality rather than a vision. Our research explores the advancements in machine learning-based object detection methods and their application to autonomous vehicle systems, specifically using the YOLO (You Only Look Once) algorithm. We begin with an overview of the YOLO algorithm, covering the YOLO architecture and comparing YOLO to alternatives like SSD and LiDAR. We then describe our case study in which we trained our own YOLO model using a custom dataset. Next, we analyze the results from the trained YOLO model to make conclusions about the YOLO algorithm. After conducting a literature review of dozens of old experiments to compare the YOLO alternatives to YOLO itself, we presented how YOLO is a better option for real-time driving scenarios in comparison to SSD (Single Shot Detector) and LiDAR (Light Detection and Ranging). In our case study of training a YOLOv8 model with our manually crafted dataset, the object detection accuracy for the model went from 20% to about 90% in only 50 epochs. We concluded that our research has highlighted YOLO's power and high speed when it comes to driverless vehicle object detection, but we also acknowledged the room available for future improvements to make roads safer.

I. Introduction

The advancement of self-driving technology has picked up speed in the last few years with a key emphasis on helping vehicles recognize and interpret their surroundings better [25]. Object detection and machine learning-based algorithms have proven to be very promising in making this vision a reality. Technological advancements have quickly improved vehicle safety by reducing the possibility of crashes involving forward collisions and steering off-lane. Furthermore, autonomous vehicles can make mobility available to a wider range of people such as the elderly, disabled individuals, and those without a driver's license. Autonomous vehicles can also increase productivity in people's lives as individuals can easily focus on important and urgent tasks, such as attending a virtual meeting or finishing an email since there is no driver, everyone is a passenger. Although some of these capabilities, such as fully autonomous operation, are already possible in limited contexts (e.g., Waymo's driverless cars in Phoenix)[1], ongoing improvements aim to enhance their reliability and scalability, allowing for these imaginations to become a reality in the near future. In autonomous driving, object detection is a key factor in ensuring safety, as it enables vehicles to identify and respond to pedestrians, vehicles on the road, and other obstacles. This paper sets out to dig deep into the trends in learning-driven object detection methods, specifically looking at how they are used in self-driving systems. Innovations in learning have brought about a variety of object detection techniques like YOLO (You Only Look Once)[1], SSD (Single Shot Detector) [2], and Faster R-CNN (Faster Region based Convolutional Neural Networks) [3]. These methods have showcased precision and the ability to process data in real-time effectively for applications such as autonomous driving.

[1] https://waymo.com

II. YOLO Algorithm Overview

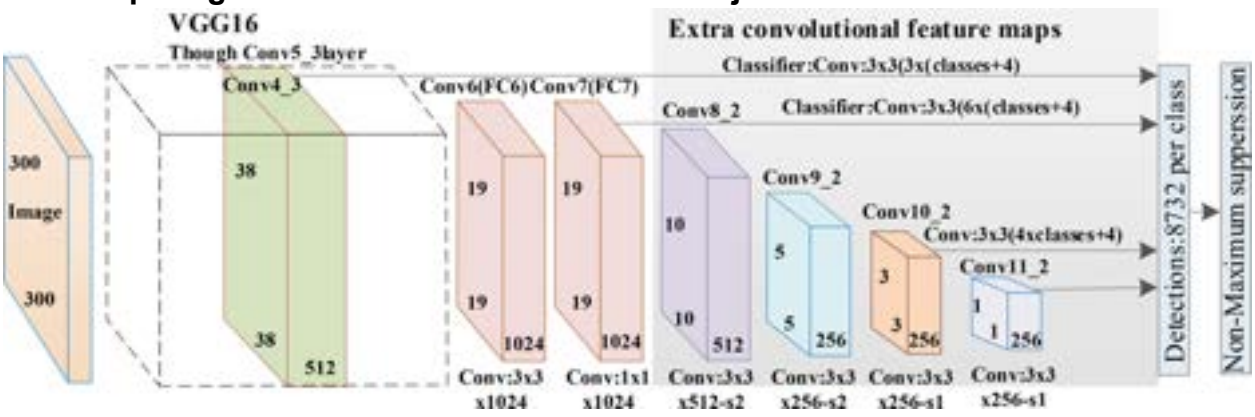


[26] Figure 1: The head, neck, and backbone make up the three primary parts of the YOLOv8 architecture. The backbone is responsible for extracting essential features from the input image using convolutional layers and C2f blocks that process image information at multiple scales, enabling effective feature representation. The neck performs multi-scale feature fusion, combining spatial and semantic information from different layers of the backbone. This is achieved through operations like upsampling and concatenation, ensuring that the model captures details critical for detecting objects of varying sizes. Finally, the head predicts bounding boxes, class probabilities, and confidence scores by processing the fused features. This

involves calculating anchor-free detection outputs that prioritize real-time performance and high precision. The entire architecture is designed for efficient object detection while balancing speed and accuracy for applications like autonomous driving.

YOLO stands for "You Only Look Once," a well-known algorithm used for detecting objects in real-time that has found extensive use in different fields like self-driving cars. It is praised for its processing and precise detection abilities, making it a top pick for real-time object identification tasks. YOLO is primarily described as a one-stage object detection framework that combines a Convolutional Neural Network (CNN) for feature extraction with real-time detection capabilities. After the YOLO framework is fed an input image into a CNN for feature extraction, the image is divided into an S × S grid, where each grid cell predicts bounding boxes, class probabilities, and a confidence score for objects present in the cell. Finally, non-maximum suppression ensures that overlapping bounding boxes are consolidated by selecting the one with the highest confidence score, thereby reducing duplicate detections [22]. The YOLO architecture, as illustrated in Figure 1, is designed to be efficient in object detection. The modular design reduces redundancy through a combination of convolutional, downsampling, and upsampling layers, followed by detection layers that can predict bounding boxes, class probabilities, and confidence scores [24]. Thus, YOLO can conduct high-speed inferences to detect in real-time. YOLO is a one-stage detector, meaning predictions are made in a single pass. Many other algorithms, such as Faster R-CNN, use two-stage detection, making them slower. YOLO has many uses other than object detection for driverless cars. It has been used for healthcare [27], agriculture [28], and even security surveillance [29]. Nevertheless, YOLO does face challenges in detecting small objects and understanding the context around them [4]. In these cases, there may be other algorithms that can be more useful, as shown in what follows.

II.A Comparing YOLO and SSD for Real-Time Object Detection



[30] Figure 2: The SSD architecture based on the VGG16 backbone (left), followed by extra convolutional layers (middle), leverages predefined anchor boxes of varying scales and aspect ratios for object detection across multiple feature layers. This design enables SSD to detect objects of different sizes but introduces added computational complexity compared to YOLO's simpler, grid-based architecture.

The architectures of SSD and YOLO neural networks consist of noticeable differences. SSD's architecture is built on the VGG (Visual Geometry Group) convolutional neural network,

which uses predefined anchor boxes at different scales and aspect ratios for object detection across multiple feature layers as seen in Figure 2 [20]. On the contrary, YOLO utilizes the Darknet convolutional neural network, a lightweight and efficient architecture that divides the input image into a grid. Each grid cell predicts bounding box coordinates, class probabilities, and confidence scores simultaneously, combining them into a unified output [21]. These distinctions between the two algorithms convey that YOLO is more suitable than SSD for real-time detection when it comes to autonomous driving. SSD's use of multiple feature layers increases its capacity to detect objects at varying sizes but comes at the cost of added computational complexity, leading to slower processing times. YOLO's single-stage design allows predictions to be made in one pass, significantly reducing latency and making it more efficient for time-sensitive applications like autonomous driving [22]. SSD is more suitable for applications that require higher accuracy for objects of unusual sizes while YOLO is preferred in time-sensitive conditions.

II.B Efficacy of YOLO vs. SSD

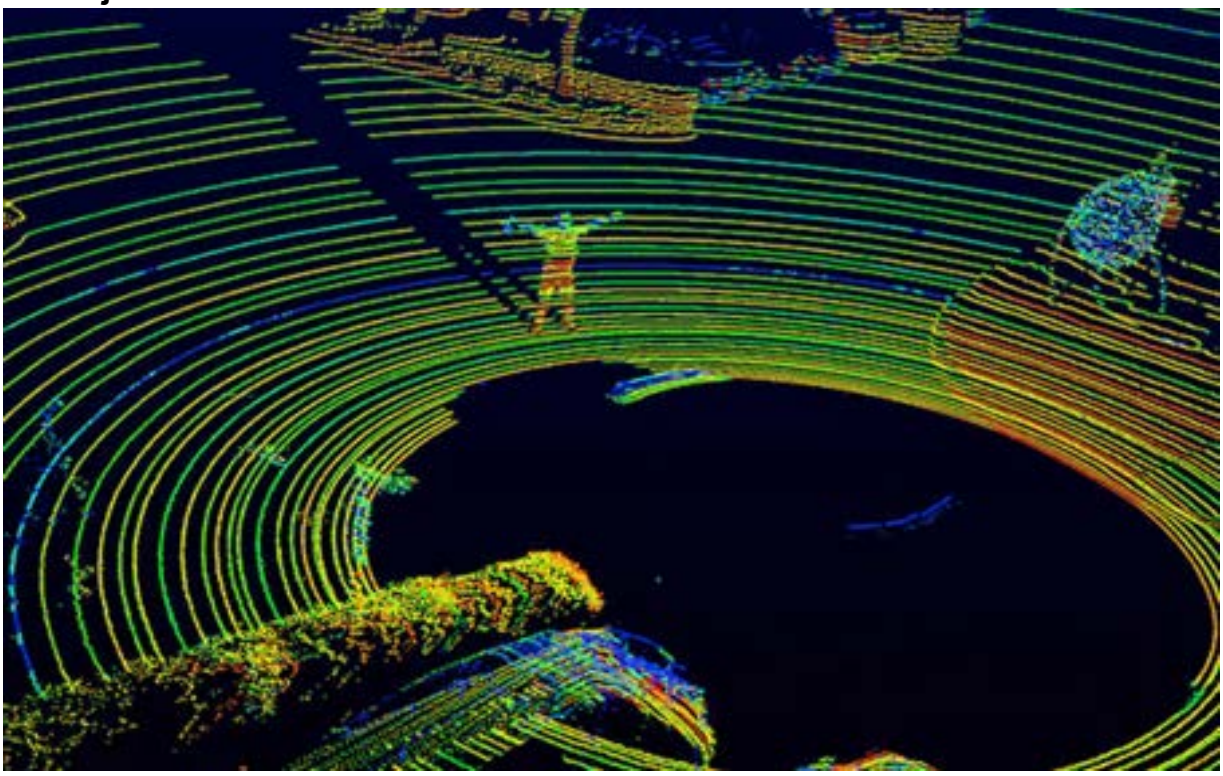
The COCO (Common Objects in Context) vehicle dataset, a subset of the COCO dataset, is commonly used to analyze the results in object detection for these algorithms. The COCO-vehicles dataset is focused on object detection in vehicles as it contains over 80,000 images of vehicles. It consists of many different vehicle categories: cars, trucks, buses, trains, motorcycles, bicycles, boats, airplanes, and ships [14]. The COCO dataset contains images of vehicles in diverse driving conditions, such as severe weather conditions, peculiar vehicle sizes, and confusing surroundings, leading to improved performance when deploying trained models in the real world.

The COCO-vehicle dataset will be used to compare the different versions of the YOLO models. YOLO models come in many different variants: YOLOv8-N(Nano), YOLOv8-S(Small), YOLOv8-M(Medium), and YOLOv8-L(Large). mAP50, which stands for Mean Average Precision at 50% IoU (Intersection over Union) threshold, is a common metric to measure the accuracy of each model's object detection capabilities when it comes to classifying objects. mAP50 is used to measure the accuracy of algorithms that are designed to classify objects. YOLOv8-N has the smallest size at about 2.5MB and has the fastest inference speed at 10ms, however, it has the lowest accuracy at mAP50: 88.2% on COCO's vehicles dataset [10]. YOLOv8-S has a slightly larger size at 12MB and a slightly higher inference speed at 30ms, but its accuracy is slightly better at mAP50: 92.5% on COCO's vehicles dataset [11]. This trend of increase in size, inference speed, and accuracy rate continues as you go closer to the large model(YOLOv8-L), which has a size of 120MB, inference speed of 80ms, and an accuracy rate of mAP50: 96.8% on COCO's vehicles dataset [12,13].

The SSD model has a 92.5% (mAP50) accuracy rate while using the COCO dataset, can detect small objects with ease, and can easily adapt to different object scales and different aspect ratios [8]. However, SSD can only detect up to 30 FPS (frames per second), which is much slower than YOLO which can detect up to 100 FPS, allowing for more accurate real-time detection. In addition, SSD has a more complex architecture than YOLO, which is has an extremely simple and efficient architecture. YOLO also has a higher accuracy rate when using the COCO dataset at 95.5%(mAP50) [9]. While SSD's multi-layer feature extraction allows it to adapt to unusual situations (e.g., construction sites, complicated road layouts) and detect small objects (e.g., motorcyclists), YOLO thrives in scenarios where speed and reaction time are prioritized [8]. Both YOLO and SSD are excellent for object detection in their own ways. When it

comes to real-time scenarios that require a quick reaction time, YOLO is more suitable because of its ability to evaluate at a staggering 100 FPS whereas SSD is only able to detect up to 30 FPS, which is significantly slower [7]. In driving, having a fast reaction time is one of the most important characteristics that a driver needs. For instance, if a car on the highway suddenly slams the brakes, the driver of the vehicle that is following behind needs to react right away to come to a stop or slow down before hitting any objects. So, YOLO would have a higher chance of crash prevention in most cases due to its faster speed, making it a better option than SSD.

II.C Object Detection with LiDAR



[19] Figure 3: An optical image taken by the Velodyne[2] LiDAR VLS-128.

Another popular alternative for YOLO is LiDAR (Light Detection and Ranging). The YOLO algorithm can be applied through the use of ordinary cameras, but LiDAR is much different. LiDAR works by emitting laser pulses that reflect off objects in the surrounding environment. The system measures the time it takes for the laser pulses to return, calculates the distance to each object, and generates a highly detailed 3D map of the surroundings known as a point cloud. As seen in Figure 3, LiDAR's ability to generate 3D point cloud data allows it to capture detailed spatial relationships in a scene with precise measurements [15]. So, LiDAR measures using exact distances, so there are extremely few chances of errors in measurements of distances. While being much more accurate than YOLO, LiDAR is much more costly than YOLO. The sensors required for LiDAR systems are much more expensive than the ordinary cameras used by YOLO. For example, a basic LiDAR sensor can cost thousands of dollars, whereas cameras used for YOLO-based vision systems are available for a fraction of that price

[2] https://www.velodyneacoustics.com/en/

[15]. One noteworthy case of opposition against LiDAR comes from a big player in the autonomous vehicles world, Tesla Motors. Tesla has opted not to use LiDAR in its self-driving systems, citing its high cost and the complexity of maintaining up-to-date mapping systems. Instead, Tesla relies on vision-based approaches, which mimic the way human drivers process their surroundings. While this perspective, advocated by CEO Elon Musk, is influential, it is not universally accepted in the autonomous driving community; instead, the main focus should be the vision as drivers also drive based on their surroundings [17]. In other words, drivers do not have all the roads around the globe memorized. The reasoning behind why YOLO is much cheaper than LiDAR is simply because YOLO uses driver point-of-view camera data and doesn't require any expensive sensors other than ordinary cameras [18]. However, if there were no constraints on budget and the main focus was reaching the most accurate measurements and detection of objects, sensor fusion could be implemented by merging LiDAR and cameras as it would make scene understanding significantly easier. Alternative algorithms like SSD and Faster R-CNN have demonstrated capabilities in specific areas, like detecting small objects and understanding context better but can sometimes lead to slower processing speeds as a trade-off [5].

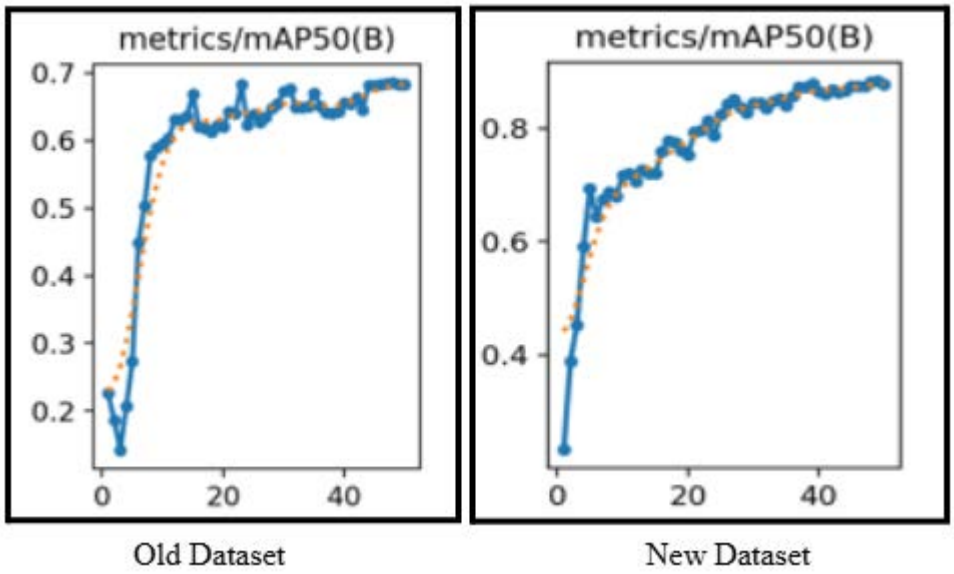
III. Case Study

In this section, we outline the process of training a YOLO model to detect various types of vehicles on the road. First, the creation of a custom dataset with over 700 images, each annotated with bounding boxes for precise object localization, is described. Next, the dataset is used to train the YOLO model, refining its ability to recognize vehicles in both urban and rural settings. Finally, the performance of the trained model is demonstrated in real-world scenarios, highlighting its capabilities in object detection and tracking across diverse road environments.

III.A Dataset

Figure 4: Batches of frames throughout the training of the YOLOv8 Model.

The custom dataset was created using Roboflow[3], a tool designed to streamline the process of building datasets for object detection. From YouTube videos, over three hours of dashcam footage was collected and uploaded into Roboflow, where each video was converted into individual frames. On each frame, bounding boxes were manually drawn, using Roboflow's labeling feature, around vehicles based on their appropriate category: cars, buses, and trucks. As seen in Figure 4, in frames with multiple vehicles this process involved adding bounding boxes for each vehicle within a single frame, ensuring that all objects were accurately labeled.



Old Dataset New Dataset

Figure 5: Accuracy in mAP50 using old dataset (left side graph) and new dataset (right side graph)

Initially, the first dataset consisted solely of footage from rural areas, with flat landscapes and minimal variation in scenery. When the model trained on this dataset was tested on city footage, its predictions were inaccurate with a mAP50 of less than 70%, as the urban surroundings introduced more complex and unfamiliar patterns. This outcome highlights a critical limitation: training a model on a dataset with uniform scenery can lead to poor generalization when tested on more diverse environments. To address this, a second, more diverse dataset was created by using video clips from YouTube to include dashcam footage featuring a variety of environments, ranging from deserts to busy cities. As seen in Figure 5, this expanded dataset significantly improved the model's accuracy in mAP50 to over 90% by exposing it to a wider range of scenarios, reducing confusion in its predictions. Figure 4 illustrates examples of bounding boxes from the dataset, showing how vehicles were annotated across different environments to enhance the model's ability to detect objects in diverse surroundings.

[3] https://roboflow.com/

III.B YOLO Model

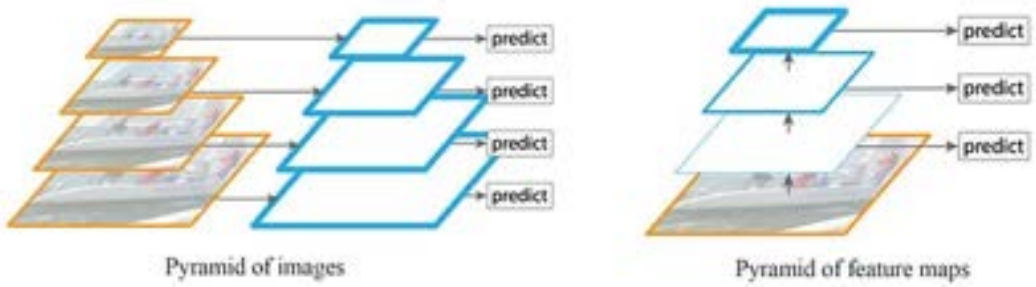


Figure 6: The pyramid of images approach (Left) resizes input images at multiple scales for prediction while the pyramid of feature maps approach (Right) efficiently uses extracted features at different layers for object detection. [23].

The YOLOv8 model architecture used here is from Ultralytics[4], a framework designed for high-performance object detection. YOLO is built on three main components: the Backbone, Neck, and Head. The Backbone, an adapted version of the Darknet[5] architecture is responsible for extracting key features from the images through several convolutional layers that downsample the input, allowing the model to pick up patterns at different scales. The Neck acts as a bridge between the Backbone and the Head, using feature pyramid layers, which are used to help detection of objects of various scales as shown in Figure 6, to enhance spatial information and ensure the model can detect objects of varying sizes. Finally, the Head generates the output by predicting bounding boxes and class probabilities for each detected object. To train this model, a custom dataset was created, and Python code was implemented to train the YOLOv8 model from scratch. The training was set to run for 50 epochs, with a batch size of 8 and an image resolution of 640 pixels.

[4] https://www.ultralytics.com/
[5] https://github.com/pjreddie/darknet

Training Code Segment:

```
from ultralytics import YOLO
model = YOLO('yolov8s.pt')

# Training
results = model.train(
   data='data.yaml',
   imgsz=640,
   epochs=50,
   batch=8,
   name='yolov8s-50e')
```

Video Prediction Code Segment:

```
from ultralytics import YOLO
# Load the trained model
model = YOLO('weights/best.pt')

# Perform inference on the video
results = model.predict(
    source='cityDriving.mp4',  # Path to the video file
    show=True,                              # Display the output in
real-time
    imgsz=1280,                             # Set the image size for
inference
    save=True,                              # Save the output
    hide_labels=True,                       # Hide labels in the output
    name='yolov8s_50e_infer1280'        # Name for the run
)

# Save the results
results.save()
```

III.C Results (Plots)

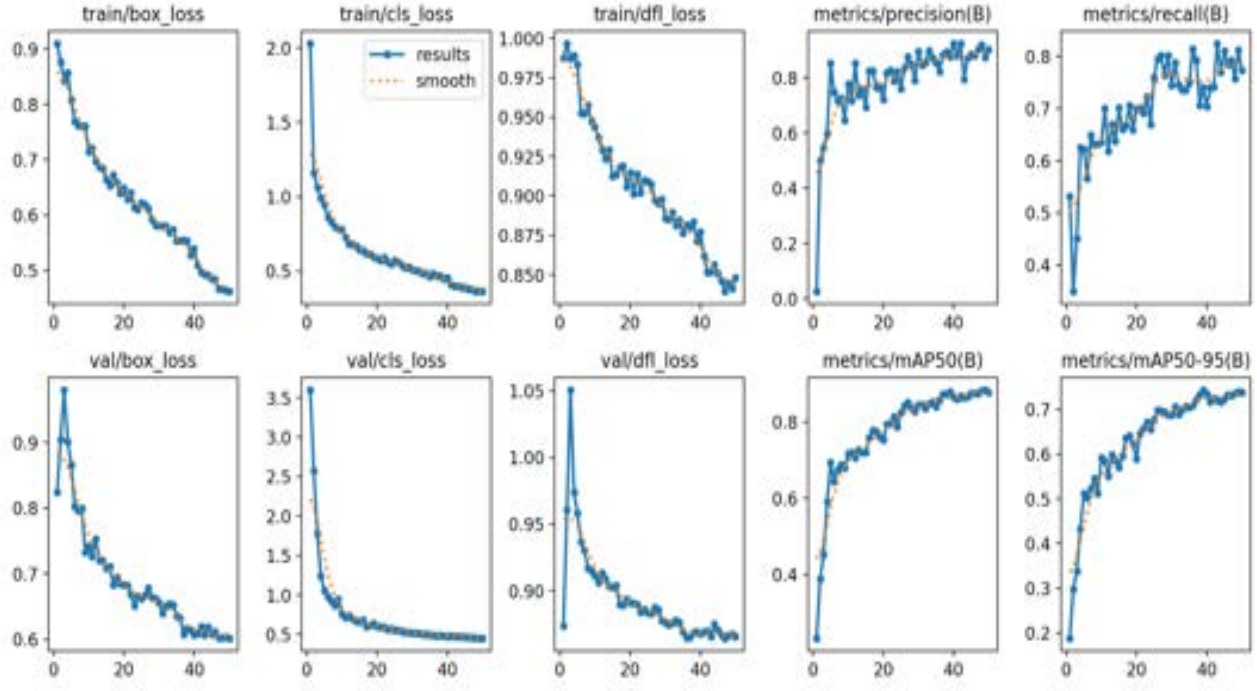


Figure 7: Results of the YOLOv8 model that we trained in 50 epochs.

Figure 7 showcases key metrics and loss trends from training a YOLO model for vehicle detection across 50 epochs. The plots reveal a steady improvement in the model's accuracy and precision, with training and validation loss for bounding boxes decreasing as the model becomes more precise in bounding box placement. Classification and DFL (Distribution Focal Loss) losses both drop, indicating the model's growing confidence in object classification and localization. Precision and recall metrics rise, showing the model's ability to detect more objects accurately while minimizing false positives and negatives. The mAP50 jumps from under 20% to about 90%, reflecting rapid learning, and the mAP50-95, Mean Average Precision averaged over IoU thresholds from 50% to 95% (at 5% steps), further demonstrates the model's adaptability to diverse object shapes. This impressive accuracy was achieved in roughly three hours on an NVIDIA RTX 3070, emphasizing YOLO's suitability for real-time object detection tasks where quick and accurate responses are essential. The results highlight YOLO's strength in performing precise object detection and tracking, making it highly effective for applications that demand on-the-spot decision-making.

Conclusion

We explored the advancements of object detection algorithms in autonomous vehicles by examining their key methods, strengths, limitations, and practical applications. After emphasizing the potential of driverless cars in the future and the critical role object detection plays in achieving the visions of driverless cars, we compared YOLO to its alternatives such as SSD and LiDAR, discussing their strengths and weaknesses. Our case study demonstrated the training process of a YOLO model using custom datasets, highlighting the importance of diverse environments in enhancing detection accuracy. The results showed YOLO's strong performance

metrics, confirming YOLO's sustainability for real-time applications. Ultimately, this exploration underscores that while YOLO excels in speed and efficiency, the choice of algorithm should align with the specific demands of the application. Our research reaffirms the importance of continual innovation in object detection as a foundation for advancing autonomous vehicle technology and improving transportation safety.

Despite making strides in the field of object detection and trajectory prediction algorithms there is still potential for enhancement. Future research avenues could involve delving into designs to enhance contextual understanding. This would involve developing algorithms capable of detecting objects with ease, even when faced with odd or cluttered backgrounds. For instance, graph neural networks can be utilized to easily interpret the relationships between objects in a scene, improving the ability to deal with complex environments. Another approach to consider is incorporating a variety of sensors, such as cameras, radar, and audio, into sensor-agnostic architectures. Separating visual and audio data before combining them could make the model more adaptable and precise in difficult situations, such as detecting vehicles in heavy fog or hearing sirens in noisy environments. Of course, having more data will not always lead to better results as at a certain point the results will stop improving. However, the results can still improve to an extent. Beyond a certain point, architecture and algorithmic improvement will become necessary to see noticeable progress in performance. The contextual understanding of algorithms can be drastically improved by enabling them to detect objects with ease despite having odd backgrounds and surroundings through the use of graph neural networks. Models can be made more robust through adversarial training. Models can be trained so that the loss function is minimized while simultaneously handling confusing scenarios, specifically distorted images or unusual object placements. Furthermore, the implementation of an active learning model would be highly beneficial because it would allow the model to learn from errors by itself similar to how humans use their mistakes to learn to do actions to prevent those mistakes in the future. The creation of such algorithms can allow self-driving vehicles to behave similarly to humans, which is crucial in making them safer because, at the moment, human judgment is much better.

References

[1] Redmon et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. CVPR.
[2] Liu et al. (2016). SSD: Single Shot MultiBox Detector. ECCV.
[3] Ren et al. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS.
[4] Redmon et al. (2017). YOLO9000: Better, Faster, Stronger. CVPR.
[5] Huang et al. (2017). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. CVPR.
[6] Lee et al. (2017). DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. CVPR.
[7] "YOLOv8: A New Era for Real-Time Object Detection" by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao (2022) - [arXiv:2207.02696]
[8] "SSD: Single Shot MultiBox Detector" by Liu et al. (2016) - [arXiv:1512.023
[9] "Object Detection Benchmarking: A Comprehensive Review" by Zhang et al. (2022) - [arXiv:2204.03590]
[10] Bochkovskiy, Alexey, et al. "YOLOv8: A New Era for Real-Time Object Detection." GitHub, 2022.
[11] Rosebrock, Adrian. "YOLOv8 Tutorial: Real-Time Object Detection." PyImageSearch, 2022.
[12] Bochkovskiy, Alexey, et al. "YOLOv8: A New Era for Real-Time Object Detection." arXiv, 2022.
[13] Kumar, et al. "A Comprehensive Review of YOLO Variants." International Journal of Computer Vision, vol. 128, no. 10, 2022, pp. 2538-2556.
[14] Lin, Tsung-Yi, et al. "Microsoft COCO: Common Objects in Context." Proceedings of the European Conference on Computer Vision, Springer, Cham, 2014, pp. 740-755, doi: 10.1007/978-3-319-10602-1_48
[15] Zhang, Y., et al. "Deep Learning for LiDAR Point Clouds: A Review." IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 1, 2022, pp. 3-14.
[16] Chen, X., et al. "LiDAR-based 3D Object Detection for Autonomous Vehicles." IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 10, 2019, pp. 2732-2742.
[17] Templeton, Brad. “Former Head of Tesla AI Explains Why They’ve Removed Sensors; Others Differ.” Forbes, 1 Nov. 2022, www.forbes.com/sites/bradtempleton/2022/10/31/former-head-of-tesla-ai-explains-why-theyve-removed-sensors-others-differ.
[18] Geiger, A., et al. "LiDAR Point Clouds for Object Detection and Tracking." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10153-10162.
[19] Murphy, Mike. “128-laser LiDAR Sensor Significantly Sharpens Autonomous Cars’ Vision.” New Atlas, 4 Dec. 2017, newatlas.com/velodyne-lidar-vls-128-sensor/52453.
[20] Liu, Wei, et al. "SSD: Single Shot MultiBox Detector." Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 21-37.
[21] Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." Journal of Computer Vision, vol. 120, no. 1, 2016, pp. 1-14.
[22] Redmon, Joseph, and Ali Farhadi. "YOLO9000: Better, Faster, Stronger." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6517-6525.
[23] Lin, Tsung-Yi, et al. Feature Pyramid Networks for Object Detection, 19 Apr. 2017, arxiv.org/pdf/1612.03144.

[24] Keita, Zoumana. “Yolo Object Detection Explained: A Beginner’s Guide.” DataCamp, DataCamp, 28 Sept. 2024, www.datacamp.com/blog/yolo-object-detection-explained.
[25] Abdullah, S. M., et al. "Self-Driving Cars: A Survey of Major Developments and Challenges." IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 4, 2022, pp. 3133-3146.
[26] Boesch, Gaudenz. “Yolov8: A Complete Guide [2025 Update].” *Viso.Ai*, 17 Oct. 2024, viso.ai/deep-learning/yolov8-guide/#:~:text=The%20neck%20merges%20these%20feature,%2Dshot%20Detector%20(SSD).
[27] Lemay, Andréanne. "Kidney Recognition in CT Using YOLOv3." arXiv, 3 Oct. 2019, https://arxiv.org/abs/1910.01268.
[28] Lawal, M.O. Tomato detection based on modified YOLOv3 framework. Sci Rep 11, 1447 (2021). https://doi.org/10.1038/s41598-021-81216-5.
[29] Ahmed, Imran et al. “A deep learning-based social distance monitoring framework for COVID-19.” Sustainable cities and society vol. 65 (2021): 102571. https://doi.org/10.1016/j.scs.2020.102571.
[30] Zhao, S., Hao, G., Zhang, Y., & Wang, S. (2021). “A real-time classification and detection method for mutton parts based on single shot multi-box detector.” *Journal of Food Process Engineering*, 44(8), e13749. https://doi.org/10.1111/jfpe.13749.