From 3D LiDAR to 2D Detection: A BEV-YOLO Approach

Technologies for Autonomous Vehicles
Politecnico di Torino

Name:	Claudia Maggiulli
Student Number:	s332252
GitHub Repository:	https://github.com/Cloud2602/BEVLidar-Object-Detection

1 Introduction

In the context of autonomous driving, LiDAR sensors play a crucial role by providing accurate 3D information about the surrounding environment. However, extracting semantic knowledge from raw point cloud data poses significant challenges, especially when attempting to perform 3D object detection directly. This task typically requires computationally expensive models and large-scale annotated datasets, which may not always be available. To address these limitations, this project proposes a 2D-based solution by converting 3D LiDAR point clouds into Bird's Eye View (BEV) images. This transformation simplifies the object detection task by projecting spatial information onto a 2D plane while preserving key geometric structures. The BEV images are then used to train a YOLO model for object detection. YOLO, known for its speed and accuracy in real-time applications, offers a practical alternative to 3D detectors in constrained environments.
The dataset used in this work is the mini version of the TruckScenes dataset, which contains synchronized LiDAR, radar, and camera data. Due to its reduced size, this dataset presents additional difficulties in achieving high performance. Various training strategies were explored, including transfer learning from pretrained YOLO models and fine-tuning with different data augmentations.
Despite these efforts, achieving strong results remains difficult, primarily due to the limited number of training samples. Nonetheless, the performance attained demonstrates the potential of the BEV-YOLO pipeline. With access to a larger and more diverse dataset in the future, it is expected that the detection accuracy and generalization capabilities of the model could be significantly improved.

2 Dataset Preparation and Labeling

2.1 TruckScenes Dataset and Sensor Modalities

The TruckScenes dataset is a multi-modal benchmark developed for perception tasks in the domain of autonomous trucks. It includes synchronized data from LiDARs, radars, and multiple RGB cameras, as well as high-quality 3D bounding box annotations. The dataset is designed to capture realistic driving conditions across a variety of urban and highway scenarios.
For this project, the **mini version** of TruckScenes was used. This reduced subset contains a smaller number of annotated frames, making it a valuable tool for rapid experimentation and prototyping. However, its limited size also introduces challenges related to overfitting and generalization.
Among all the available sensors, particular attention was given to the **LiDAR sensor**. LiDAR (Light Detection and Ranging) provides precise 3D measurements of the surrounding environment by emitting laser pulses and measuring their return times. In TruckScenes, multiple LiDAR units are mounted on the vehicle to ensure full 360-degree coverage. This allows the construction of dense point clouds, which are critical for understanding scene geometry and detecting objects with high spatial resolution.
In this work, LiDAR data was the primary source used for creating Bird's Eye View (BEV) images. These top-down representations enable effective 2D object detection by preserving the spatial arrangement of objects while reducing the dimensionality of the problem. The focus on LiDAR data was motivated by its robustness in varied lighting and weather conditions, which makes it especially suited for autonomous driving applications.

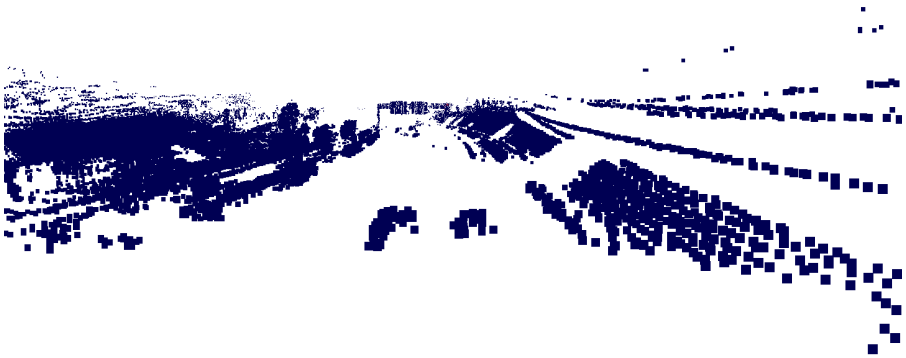

Figure 1: Rendering Lidar 3D

2.2 From 3D LiDAR to BEV Representation

The 3D point cloud data provided in the TruckScenes mini dataset was transformed into 2D Bird's Eye View (BEV) images to enable object detection using 2D convolutional models like YOLO. This transformation was conducted through a custom pipeline developed in Python, leveraging the dataset's official development kit. The pipeline starts by loading raw point clouds from all available LiDAR sensors on the vehicle, resulting in a fused and comprehensive 360-degree spatial representation. The point clouds are extracted using the `from_file_multisweep()` function, which combines multiple temporal sweeps if necessary to densify the cloud. Each point cloud is then projected onto a flat 2D plane using transformation matrices derived from the vehicle's ego-pose and sensor calibration data. This projection is implemented through the composition of several transformations: from sensor to ego vehicle coordinates, and from ego vehicle to a top-down, "flat" view aligned with the ground plane. The `view_points()` utility is then used to project the 3D points onto the 2D BEV frame.

The point cloud intensity values are normalized and mapped to color channels using a perceptually optimized colormap, enhancing contrast and aiding object visibility. To improve compatibility with YOLO, the output resolution of the BEV images was set to 1024×1024 pixels and saved in PNG format.

Finally, to enlarge the dataset and increase its variability, each BEV image was generated twice: once using the full point cloud, and once with the application of a z-filter to exclude low-density or high-elevation background points. These dual representations provide diversity in appearance and object visibility, which proved beneficial for training deep learning models.

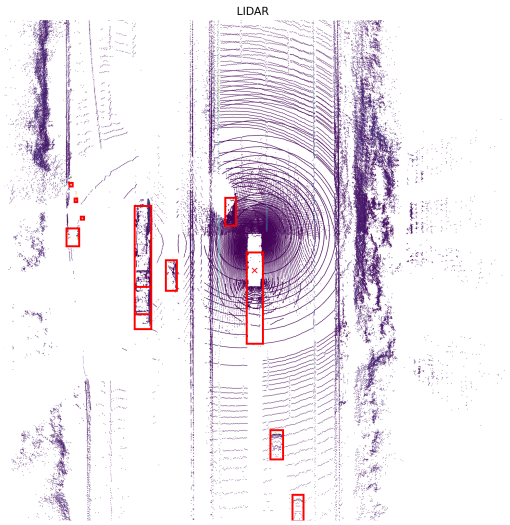

Figure 2: Example of dataset BEV images with bounding boxes

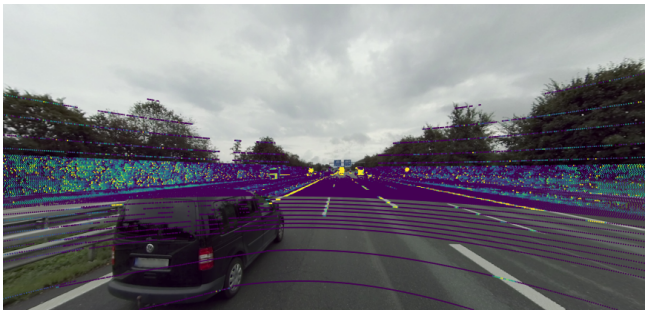

Figure 3: Plot the radar point cloud on image

2.3 Label Generation and Class Simplification

To train YOLO with BEV images, 3D annotated bounding boxes provided in the TruckScenes dataset were converted into 2D image annotations. The process begins by projecting the corners of each 3D bounding box onto the Bird's Eye View plane using the same transformation matrices applied to the LiDAR point cloud.

For each projected box, the center coordinates, width, and height were computed in meters. These values were then normalized relative to the image size (e.g., 80 m $\times$ 80 m scene mapped to 1024 $\times$ 1024 pixels), resulting in YOLO-compatible annotations in the format `<class_id> <x_center> <y_center> <width> <height>`, all within the range $[0, 1]$.

Given the large number of fine-grained classes in the original dataset (e.g., `vehicle.truck`, `vehicle.bus.rigid`, `vehicle.trailer`), a label reduction strategy was applied to simplify the classification task. Similar or semantically overlapping categories were grouped together. For example, all heavy vehicles such as `truck`, `bus.rigid`, and `trailer` were mapped to a single class named `vehicle.heavy`. Pedestrians of different types (e.g., `adult`, `child`, `construction_worker`) were also merged into a unified `human.pedestrian` class.

This mapping was implemented via a lookup table that transformed each original class ID into a new reduced one. All YOLO label files were automatically converted based on this mapping, significantly decreasing class imbalance and making the learning task more tractable with a small dataset.

3 Model Architecture and Training Pipeline

3.1 YOLOv8 Overview

YOLOv8 is the latest release in the "You Only Look Once" family of real-time object detectors, developed by Ultralytics. It introduces several architectural improvements and training optimizations compared to its predecessors. Among the key features are an anchor-free detection head, dynamic input resizing, and support for multiple tasks including object detection, segmentation, and classification.

The model family includes variants of increasing size and complexity (`n`, `s`, `m`, `l`, `x`), enabling flexible deployment based on available computational resources. Due to computational constraints in this work, only the `n`, `s`, and `m` versions were tested.

Training and inference were performed using the official `ultralytics` Python package, which offers a streamlined API and built-in support for logging, augmentation, and model export.

3.2 Pretraining Strategy (Frozen Backbone)

In the initial experiments, the full YOLOv8 architecture was trained end-to-end on the custom BEV dataset. However, due to the limited size of the dataset and the risk of overfitting, a revised strategy was adopted: the backbone of the network was frozen during the early training phase by setting `freeze=10` in the training pipeline.

This approach prevents the pretrained feature extractor from being updated while allowing the detection head to adapt to the new domain. Since the BEV images derived from LiDAR data differ significantly from the RGB images used for YOLO's original pretraining (typically on datasets like COCO), retaining general visual features while specializing the detector proved to be beneficial.

Freezing the backbone leverages the robustness of the pretrained features while reducing the number of trainable parameters, making the training process more stable and less prone to overfitting—an important consideration given the relatively small size and domain-specific nature of the dataset.

3.3 Fine-Tuning and Augmentation

After the initial pretraining phase, the entire network was fine-tuned to adapt more closely to the BEV representation of the LiDAR data. Fine-tuning allowed the model to update all its parameters, improving its ability to capture the specific spatial patterns of the dataset. To further enhance generalization and robustness, a set of lightweight data augmentation techniques was employed. These included small affine transformations such as rotations, translations, scaling, and shearing, which help the model become invariant to slight geometric variations often encountered in real-world autonomous driving scenarios. Label smoothing was also adopted to prevent the network from becoming overly confident in its predictions. These regularization strategies proved essential, especially given the limited size and domain-specific nature of the dataset.

3.4 Evaluation Metrics

The performance of the trained models was assessed using standard object detection metrics, namely *mean Average Precision* at different thresholds and classification metrics such as *precision* and *recall*. The **mAP@0.5** metric evaluates detection performance using an Intersection over Union (IoU) threshold of 0.5, rewarding correct detections with moderate overlap. The more stringent **mAP@0.5:0.95** (also referred to as mAP50–90) averages the precision across multiple IoU thresholds ranging from 0.5 to 0.95 in steps of 0.05, providing a more comprehensive measure of both localization and classification quality. **Precision** indicates the proportion of correctly predicted objects among all predicted objects, while **recall** measures the ability to correctly detect all ground truth objects. Together, these metrics offer a robust evaluation of the model's accuracy, sensitivity, and overall detection quality.

4 Experiments and Results

4.1 Training Settings and Hyperparameters

To evaluate the performance of YOLOv8 on BEV (Bird's Eye View) LiDAR images, three progressively more advanced training configurations were implemented. The first two configurations were trained on the original dataset composed of 400 BEV images, while the third configuration was trained on an enlarged dataset containing 800 BEV images, including additional samples obtained through a Z-axis height filter to enhance object visibility and reduce background noise.
Each configuration introduces specific improvements in terms of model complexity, input resolution, training strategy, and data augmentation, with the aim of incrementally improving detection accuracy and generalization.

Configuration 1 – Baseline Training with YOLOv8n. The first configuration uses YOLOv8n, the smallest model variant in the YOLOv8 family, to establish a computationally lightweight baseline. It is trained on the standard dataset of 400 BEV images, using a resolution of 640×640 pixels and a batch size of 16. This setup allows fast training and serves as a benchmark for more complex strategies.

- **Model:** YOLOv8n
- **Epochs:** 50
- **Image size:** 640
- **Batch size:** 16
- **Augmentations:** default (minimal)

Configuration 2 – Pretraining and Fine-tuning with YOLOv8s. In the second configuration, a more expressive model (YOLOv8s) is used, together with a two-stage training procedure. Like the first, it is trained on the standard dataset of 400 BEV images. The input resolution is increased to 1024×1024 to retain finer spatial details in BEV projections, and training is split into a 40-epoch pretraining phase and a 20-epoch fine-tuning phase.
During pretraining, the first 10 layers of the backbone are frozen in order to retain previously learned low-level features while focusing on adapting higher-level layers to the BEV input domain. The batch size is reduced to 8 due to memory requirements, and cosine learning rate scheduling with warmup and weight decay is used to ensure stable convergence. No augmentations are applied in this configuration to isolate the effect of model size and training strategy.

- **Model:** YOLOv8s
- **Dataset:** standard (400 BEV images)
- **Pretraining:**
  - Epochs: 40, Image size: 1024, Batch size: 8
  - Learning rate: 0.001, Warmup: 3 epochs
  - Weight decay: 0.0005, Cosine LR scheduling: enabled
  - Frozen layers: 10 (partial backbone freezing)
- **Fine-tuning:**
  - Epochs: 20, Learning rate: 0.0001, Warmup: 2 epochs

Configuration 3 – Augmented Training on an Enlarged Dataset with YOLOv8m. The third configuration introduces several improvements. It uses the larger YOLOv8m model and is trained on an enlarged dataset composed of 800 BEV images, which includes both the standard samples and additional Z-filtered images that help suppress ground clutter and highlight elevated objects.
The training again follows a two-stage approach. During pretraining, the backbone is partially frozen to focus learning on the detection heads. In the fine-tuning stage, a range of geometric augmentations (e.g., rotation, scaling, shear, translation, mosaic) is applied to increase robustness. Label smoothing is also used throughout to prevent overconfidence in the model's predictions.

- **Model:** YOLOv8m
- **Pretraining:**
  - Epochs: 50, Image size: 1024, Batch size: 8
  - Freeze: 10 backbone layers, Learning rate: 0.001
  - Label smoothing: 0.01
- **Fine-tuning:**
  - Epochs: 40, Learning rate: 0.001
  - Augmentations:
    - Rotation: $\pm 1°$, Scaling: $\pm 5\%$
    - Shear: $\pm 1°$, Translation: $\pm 5\%$
    - Mosaic: 0.05
  - Label smoothing: 0.01

These three configurations reflect a progressive refinement strategy, transitioning from a basic setup to a more advanced pipeline with larger models, richer data, and targeted regularization techniques.

4.2 Training Comparison

The following table compares the performance of the three training configurations using standard evaluation metrics: mAP_{50}, $\text{mAP}_{50:95}$, Precision, and Recall.

Configuration	**mAP_{50}**	**$\text{mAP}_{50:95}$**	**Precision**	**Recall**
YOLOv8n (Baseline)	14%	10%	16%	15%
YOLOv8s (Pretrain + Fine-tune)	40%	37%	43%	35%
YOLOv8m (Augmented + Z-filtered)	88%	71%	92%	85%

Table 1: Performance comparison of the three YOLOv8 training configurations on BEV LiDAR data.

Performance Analysis. The results clearly show a progressive improvement across the three configurations. The baseline model, YOLOv8n, achieved only modest performance, with an mAP_{50} of 14% and a low recall of 15%. This is expected given its limited capacity and lower input resolution, making it unsuitable for capturing the complex spatial structures present in BEV LiDAR data.
The second configuration, using YOLOv8s with pretraining and fine-tuning, significantly improves all metrics. The mAP_{50} rises to 40%, and $\text{mAP}_{50:95}$ to 37%, indicating better localization and class confidence. Precision and recall also show noticeable gains, confirming the effectiveness of a two-phase training strategy and increased input resolution.
The third configuration, based on YOLOv8m trained on the enlarged dataset with data augmentations, achieves the best results by a large margin. An mAP_{50} of 88% and a precision of 92% highlight its strong detection capability, while the recall of 85% confirms high sensitivity to true positives. The inclusion of Z-filtered images and geometric augmentations, combined with a deeper model and regularization techniques like label smoothing, clearly contributes to superior generalization and robustness.
Overall, these results demonstrate that increasing model complexity, input richness, and applying structured training strategies have a substantial impact on detection performance in the BEV domain.

4.3 Confusion Matrix and Bounding Box Analysis of the Best Configuration

To better understand the behavior of the best-performing model (YOLOv8m, trained on the enlarged dataset with augmentation), a deeper analysis was performed using the confusion matrix, per-class distribution, and training curves.

Confusion Matrix Analysis. The confusion matrix (See Appendix 4) confirms that the model exhibits strong class discrimination, particularly for high-frequency classes such as `vehicle.car` and `vehicle.heavy`, with 516 and 273 correctly classified instances, respectively. Some confusion is still present for visually or spatially similar classes, such as `static_object.traffic_sign` and `movable_object.barrier`, or between `vehicle.bicycle` and `human.pedestrian`, which often appear in close proximity in BEV images. The class `animal` shows very few detections and misclassifications, which is expected given its low frequency in the dataset.

Instance and Per-Box Distribution. As shown in the Appendix 5, the distribution of instances is highly imbalanced, with classes like `vehicle.car`, `static_object.traffic_sign`, and `vehicle.heavy` dominating the dataset. Rare classes such as `vehicle.train`, `animal`, and `vehicle.other` have very few instances, which likely impacts detection performance for these categories. The per-box statistics in normalized coordinates confirm that most bounding boxes are concentrated in the central part of the image (x 0.5, y 0.5) and tend to be narrow and short in height, typical of distant BEV projections.

Training Curves. The training curves in Appendix 6 show a consistent and smooth convergence. All loss components (box, classification, and distribution focal loss) decrease steadily during training and validation. Precision and recall improve significantly over the epochs, stabilizing around 0.92 and 0.85 respectively. The mAP_{50} and $mAP_{50:95}$ metrics also exhibit a positive trend, ultimately reaching 88% and 71%, confirming the model's strong generalization and detection capabilities.

5 Conclusion

This work has demonstrated the effectiveness of using 2D Bird's Eye View (BEV) projections as a representation for processing 3D LiDAR point clouds in the context of object detection. By converting complex 3D spatial information into structured 2D images, it becomes possible to leverage state-of-the-art convolutional neural networks, such as YOLOv8, to perform real-time detection with high accuracy and robustness.
The results obtained on the TruckScenes dataset confirm the validity of this approach. The best-performing configuration, based on YOLOv8m trained on an augmented BEV dataset with Z-filtered images, achieved excellent detection scores: mAP_{50} of 88%, $mAP_{50:95}$ of 71%, with a precision of 92% and a recall of 85%. These results show that BEV-based 2D detection pipelines can effectively retain essential spatial information from LiDAR and generalize well to diverse object categories, particularly those with sufficient representation in the training set.
Despite these promising outcomes, challenges remain. Several object classes, such as `animal`, `vehicle.train`, and `vehicle.other`, exhibited significantly lower detection performance, primarily due to data imbalance and limited instance frequency. Future work should explore strategies such as:

- Training on larger and more balanced datasets with better coverage of rare classes;
- Implementing class-aware sampling or synthetic data augmentation;
- Integrating multi-modal inputs (e.g., camera + LiDAR) or incorporating temporal information from sequences;
- Exploring hybrid approaches that combine BEV with 3D-centric architectures for higher-fidelity modeling.

In conclusion, BEV-based 2D methods offer a scalable and efficient pathway for deploying LiDAR-based perception systems in real-world autonomous driving scenarios, and they represent a solid foundation for further exploration in more complex and diverse environments.

Appendix

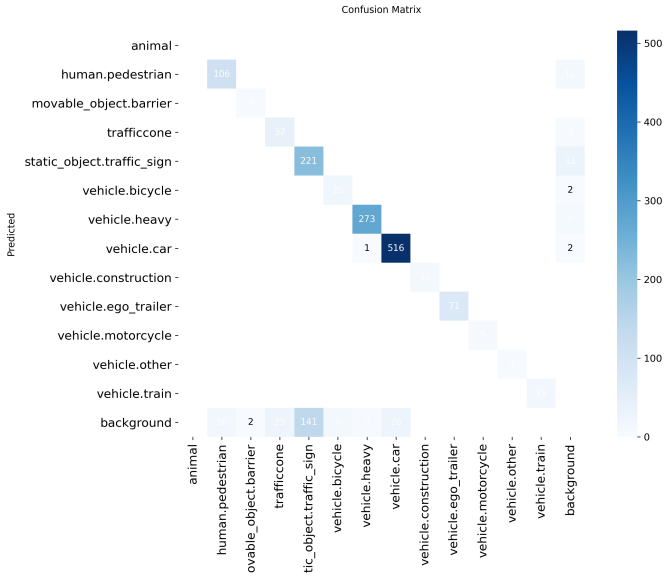

Figure 4: Confusion matrix of the best model (YOLOv8m).

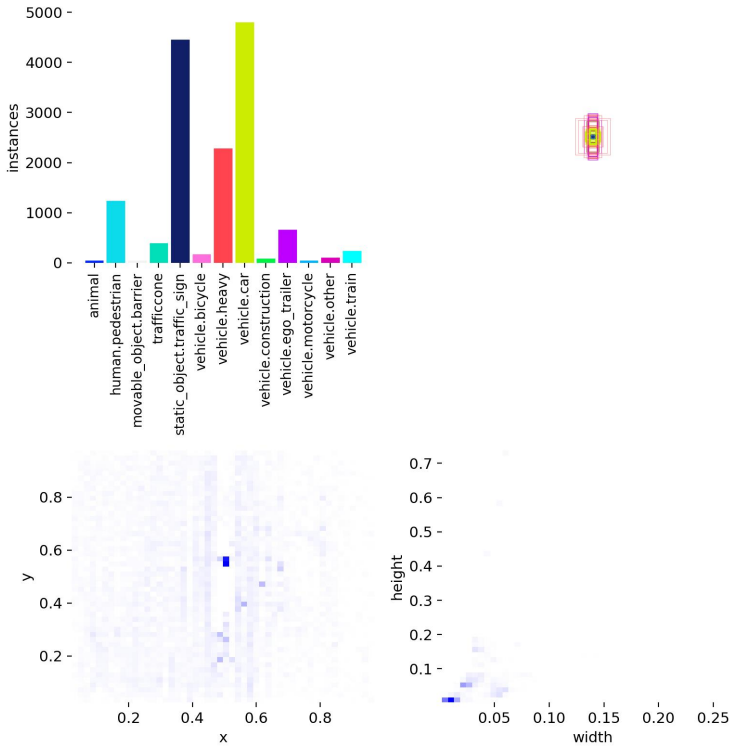

Figure 5: Instance distribution and bounding box statistics.

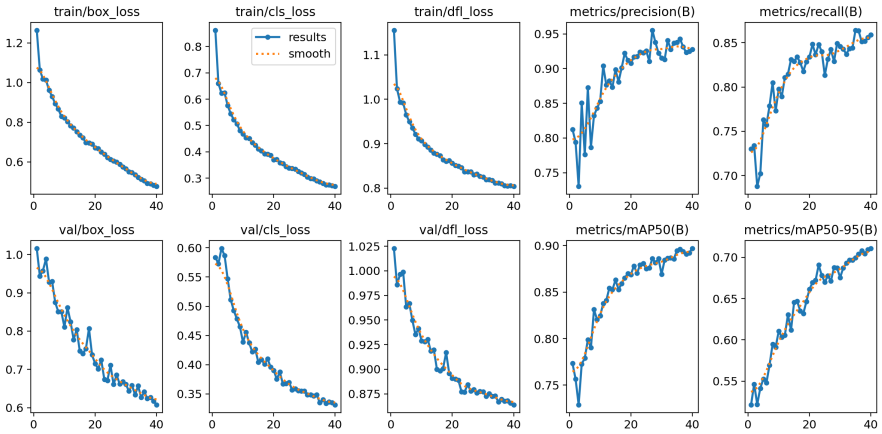

Figure 6: Training and validation metrics.

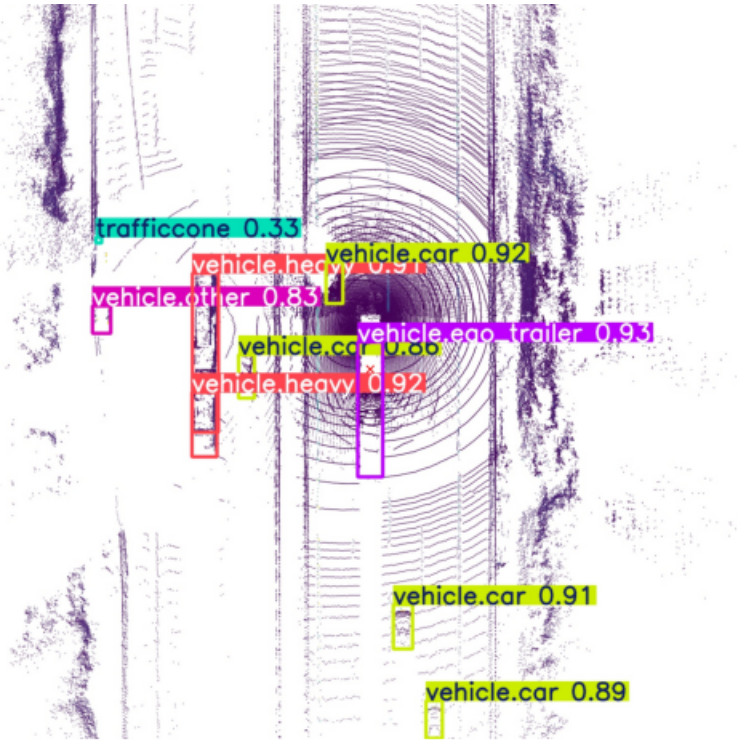

Figure 7: Example of a prediction.

References

References

[1] TUM FTM, *TruckScenes Devkit*, https://github.com/TUMFTM/truckscenes-devkit.

[2] F. Fent, F. Kuttenreich, F. Ruch, F. Rizwin, S. Juergens, L. Lechermann, C. Nissler, A. Perl, U. Voll, M. Yan, and M. Lienkamp, *MAN TruckScenes: A multimodal dataset for autonomous trucking in diverse conditions*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2022.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] B. Yang, W. Luo, and R. Urtasun, *PIXOR: Real-time 3D Object Detection from Point Clouds*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.