TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Anomaly Detection for Semi-Supervised 3D Object Detection

Friedrich Dang

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Anomaly Detection for Semi-Supervised 3D Object Detection

Anomaliedetektion für halbüberwachte 3D Objektdetektionen

Author:	Friedrich Dang
Supervisor:	Prof. Dr.-Ing. Matthias Althoff
Advisor:	Tobias Ladner, M.Sc.
Submission Date:	26.01.2025

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Munich, 26.01.2025 Friedrich Dang

Acknowledgments

I would like to express my gratitude to SetLabs and the Department for Cyber Physical Systems at the Technical University of Munich (TUM) for providing the necessary resources and support throughout the duration of this project. Their contributions were instrumental in enabling this research.

A special thanks to Mr. Max Ronecker (SetLabs) and Mr. Tobias Ladner (TUM) for their invaluable guidance, insightful feedback, and encouragement. Their expertise and mentorship have significantly shaped the direction and outcomes of this work.

Abstract

Accurate 3D object detection is critical for autonomous driving, yet manually labeling large-scale LiDAR datasets remains a resource-intensive process. While existing automated labeling methods offer potential solutions, they are often hindered by high computational demands, dependence on extensive labeled data, and limited effectiveness for rare or underrepresented object classes.

To address these challenges, we propose a modular framework that refines object detections and reduces false positives by focusing on anomaly detection. Leveraging a track-centric data structure, our approach filters incorrect predictions while retaining valid ones. Evaluated on the nuScenes dataset, the framework significantly improves detection performance, particularly for rare object classes, in both fully supervised and semi-supervised settings. These results demonstrate the potential of scalable and lightweight frameworks to enhance automated 3D labeling while tackling real-world data constraints in autonomous driving.

Contents

1. Introduction

Imagine you are tasked with labeling data to support your team in creating a new 3D dataset for autonomous driving, aimed at improving object detection models. After annotating a few hundred samples, such as drawing 3D bounding boxes around vehicles, pedestrians, and other objects in LiDAR point clouds, it becomes clear that the process is repetitive, resource-intensive, and time-consuming. You begin to wonder if there is a more efficient way - possibly a machine learning model that could automate parts of this workload. Such a model could allow your team to focus on refining the detection pipeline or tackling other higher-level challenges, like improving prediction accuracy or robustness under edge cases.

Building a high-quality dataset for training and evaluating neural networks, particularly in 3D object detection, is a challenging endeavor. It demands precise annotations, robust validation processes, and significant resources. These challenges are magnified when dealing with 3D data due to its inherent complexity, such as managing spatial relationships and multiple viewpoints. For smaller organizations, the time and cost associated with manual annotation can be a significant barrier, limiting the feasibility of such projects, as a study by Dimensional Research [44, 52] estimates that annotating 100,000 samples can take 300 to 850 hours, highlighting the substantial resources needed for this process.

To address these challenges, the CTRL system [12] introduced a LiDAR-based auto-labeling approach that achieved state-of-the-art performance on the Waymo Open Dataset (WOD). CTRL demonstrated its ability to outperform human annotators in some scenarios, identifying 99.52% from 1.03M selected vehicle objects. However, this success came with substantial resource requirements, including the use of 8 GPUs for 20 hours of for training per class [10] and a large proportion of labeled data. For example, CTRL was evaluated using 70% of WOD for training and 30% for validation. While this configuration yielded impressive results, it remains unclear how the system would perform with less annotated training data, making its accessibility for resource-constrained teams an open question.

Additionally, systems like CTRL often optimize for mean average precision (mAP), favoring high recall to maximize their performance metrics. While this strategy increases the total number of objects detected, it also results in a large number of false positives, which must be reviewed and corrected manually. This trade-off shifts the workload from labeling to verification, highlighting a gap in current approaches that could benefit from refinement.

This work proposes a novel solution to address these challenges by incorporating anomaly detection into the labeling pipeline (Figure 1.1). By focusing on identifying and

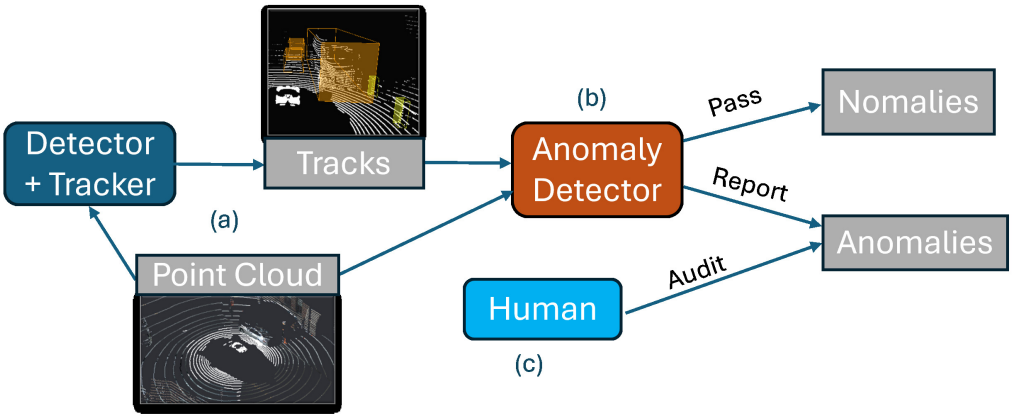


Figure 1.1.: Overall idea: Starting from a point cloud input (a), an off-the-shelf object detector and tracker generate detection tracks. These tracks are then processed by an anomaly detector (b) to classify tracks' detections as "nomalies" and "anomalies". Finally, a human annotator (c) reviews only the detected anomalies, significantly reducing the overall annotation effort.

filtering false positives after generating high-recall predictions with the baseline model, the proposed approach improves overall precision and aims to reduce the workload for human reviewers. Additionally, the anomaly detection framework provides insights into the mistakes made by the baseline model, guiding reviewers toward areas requiring attention and supporting the quality control process.

To the best of our knowledge, this is the first study to apply anomaly detection techniques in 3D object detection to improve the efficiency and accuracy of data labeling. While the results of this project are limited, they demonstrate the potential of a semi-supervised framework in reducing annotation costs. Specifically, our method achieved better results compared to the baseline model, CenterPoint, when trained on only 5% of the training data. Furthermore, these improvements were achieved using a single GPU, underscoring the efficiency of the proposed approach.

With our research, we aim to contribute to the broader field of dataset creation for machine learning by exploring a cost-effective solution that reduces reliance on extensive manual annotation while maintaining high-quality results.

2. Fundamentals

This thesis explores computer vision and deep learning, focusing on 3D object detection and tracking, which are crucial for autonomous driving systems. In this chapter, we introduce evaluation metrics, our used loss functions, base detector and tracker, CenterPoint [60] and ImmortalTracker [55], highlighting their roles as the backbone of our approach. We also discuss prominent 3D datasets for autonomous driving, the annotation efforts associated with them.

2.1. Evaluation Metrics - Binary Classification

In binary classification, there are typically two possible outcomes: positive or negative. Each prediction can either match the actual class (true) or differ from it (false), leading to four possible scenarios: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). For instance (Figure 2.1), if a model designed to classify cars and pedestrians in autonomous driving incorrectly classifies a car as a pedestrian, it would result in a false positive [43].

Building on this framework, the following section introduces the key evaluation metrics used in this study, starting with accuracy. Accuracy is a fundamental metric that evaluates the overall performance of the model by calculating the proportion of correct predictions out of the total number of predictions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2.1}$$

2.1.1. Recall, True Negative Rate, and Balanced Accuracy

Recall and True Negative Rate (TNR) are fundamental metrics, each emphasizing different aspects of model performance. Recall, also known as the True Positive Rate (TPR), measures the proportion of actual positive instances that are correctly identified by the model. High Recall indicates the model's effectiveness in detecting positive cases, making it crucial in scenarios where missing a positive instance, such as in fraud detection, carries significant consequences. It is defined as [43]:

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.2}$$

In contrast, the True Negative Rate quantifies the model's ability to correctly identify negative instances. A high TNR is essential in applications where minimizing false

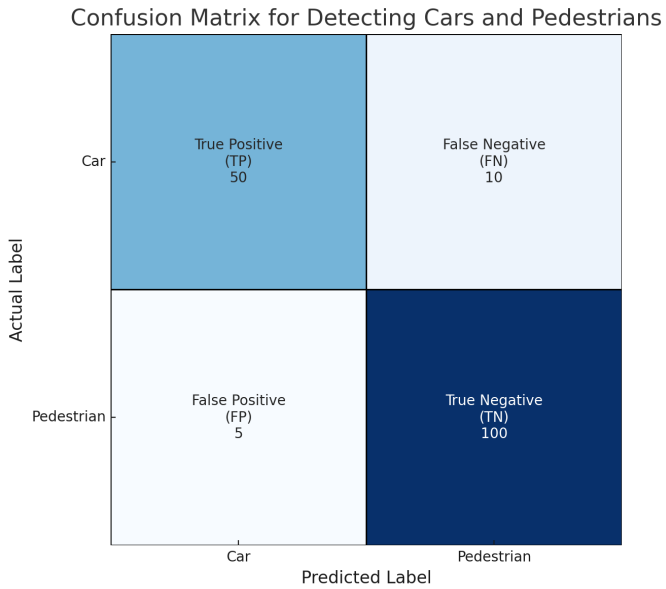


Figure 2.1.: An example of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) in the context of classifying cars and pedestrians, illustrated using a confusion matrix. The accuracy of the classification model is approximately 90.91%.

alarms, such as in fraud detection, is a priority. It is calculated as [43]:

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{2.3}$$

Together, Recall and TNR provide complementary insights into a model's performance, highlighting its capability to correctly classify both positive and negative instances while accounting for different types of errors. Calculating the mean of these two metrics results in Balanced Accuracy, which is especially insightful for imbalanced datasets as it equally weighs the performance on both classes, preventing dominance of the majority class in the evaluation.

2.1.2. False Positive Rate and False Negative Rate

False Positive Rate (FPR) and False Negative Rate (FNR) are closely related to the above introduced True Negative Rate (TNR) and True Positive Rate (TPR), respectively. In particular, $\text{FPR} = 1 - \text{TNR}$ and $\text{FNR} = 1 - \text{TPR}$. The FPR quantifies the proportion of actual negative instances that are incorrectly classified as positive, capturing how often the model produces false alarms [43]:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{2.4}$$

where FP and TN denote the number of false positives and true negatives, respectively.

Conversely, the FNR measures the proportion of actual positive instances that the model fails to identify correctly:

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}'}, \tag{2.5}$$

where FN and TP represent the number of false negatives and true positives, respectively. By considering FPR and FNR alongside TNR and TPR, one gains a complete picture of the model's strengths and weaknesses in both identifying true positives and correctly rejecting negatives [43].

2.1.3. Precision, Average Precision, Mean Average Precision

Precision and Mean Average Precision (mAP) are essential metrics for evaluating multi-class classification tasks. Precision measures the proportion of predicted positive instances that are actually correct. High Precision indicates that when the model predicts a positive outcome, it is likely correct. This is particularly important in applications, where false positives can have significant costs such as medical testing. It is calculated as [43]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.6}$$

Average Precision (AP) is a metric commonly used in object detection tasks to evaluate the performance of a model. In detection tasks, predictions are ranked by confidence, meaning that the model's predictions are sorted based on how confident it is about each prediction, from the highest to the lowest confidence. For each confidence-ranked prediction, an instance of Precision and Recall are calculated using the corresponding TP, FP, FN, based on a distance metric relative to the ground truth [11].

AP is then calculated by averaging the maximum precision p values across n equally spaced Recall levels $\tilde{r}$:

$$\text{AP} = \frac{1}{n} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r}). \tag{2.7}$$

The final AP score provides a single value that summarizes the model's ability to rank positive instances highly while minimizing false positives [11].

On the other hand, Mean Average Precision (mAP) is a more comprehensive metric used in tasks where multiple classes or multiple thresholds are involved. It is the mean of Average Precision values calculated for each category, defined as [6]:

$$\text{mAP} = \frac{1}{m} \sum_{i=1}^{m} \text{AP}_i, \tag{2.8}$$

where AP_i is the average Precision for classes i, and m is the total number of classes. mAP provides a balanced evaluation of model performance across all classes. For different benchmarks, different versions of means of average Precision are employed e.g., in (2.2.1, 2.2.2).

2.2. Datasets for Autonomous Driving

The key source for any machine learning task is its dataset. Critical aspects of such datasets are scale, diversity and modality. In the following, we present two popular, large-scale datasets, used for our research: the Waymo Open Dataset [48] and nuScenes dataset [6]. Most of our works are based on nuScenes (example frame in Figure 2.2).

2.2.1. Waymo Open Dataset

The Waymo Open Dataset [48] comprises diverse real-world data collected from varied urban, suburban, and highway environments. Currently, it is the largest autonomous driving dataset by volume with over 20 million frames of LiDAR, and camera data. It provides exhaustive annotations at 10 Hz for vehicles, pedestrians, cyclists, and street signs, with over 10 million 3D bounding boxes across a variety of weather and lighting scenarios.

The autonomous driving company Waymo [56] has hosted several public challenges based on its published dataset, enabling the benchmarking of models for 3D detection and tracking. From the 1.2k driving sequences, 798 are used for training, 202 validation, and 200 for testing. For evaluation purposes, the annotations of the test set are withheld. Their official 3D detection evaluation metrics include mean average precision (mAP) and mAP weighted by heading accuracy (mAPH). Both performance metrices are based on class dependent intersection over union (IoU) thresholds and are further divided into two difficulty levels: Level 1 (with at least five LiDAR points in the 3D bounding boxes) and Level 2 (with at least one LiDAR point).

2.2.2. nuScenes Dataset

The nuScenes dataset [6] is widely used for research in autonomous driving, offering a diverse and multimodal perspective on challenging urban driving scenarios in Boston and Singapore. While nuScenes provides in total fewer annotations than WOD, primarily due to annotating at 2Hz instead of 10Hz, annotations cover a more diverse set of ten categories in a long-tail distribution, including vehicle trailers, traffic cones and motorcycles.

The dataset contains 1k driving sequences, with 700, 150, and 150 sequences for training, validation and testing, with 28k, 6k, and 6k annotated frames, respectively. Similar to Waymo's perception challenges, Motional [36] conducts challenges using their nuScenes dataset, withholding the annotations for the testing sequences. For

highly accessible evaluation and development purposes, there is a provided nuScenes-devkit [14].

The official evaluation metrics for 3D object detection are mAP, a nuScenes detection score and a neural planning metric (PKL). Here, the mAP is an average Precision among the classes and the mean of four distance thresholds (0.5m, 1m, 2m, 4m) to the object's center in bird's-eye-view [6].

NuScenes detection score is a weighted average of mAP and other metric errors, including velocity and box attributes [38]. The PKL [38] metric uses predicted 3D detections and evaluates their impact for down-streamed planning tasks measuring the KL divergence of a planner's trajectory and ground truth trajectories.

2.3. Binary Cross-Entropy Loss & Focal Loss

Binary Cross-Entropy Loss is a common loss function in binary classification tasks and Focal Loss a typical extension of loss functions in scenarios where the class distribution is imbalanced. Binary Cross-Entropy Loss (BCE) penalizing incorrect predictions by measuring the difference between the predicted probabilities and the actual labels [33]:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right], \tag{2.9}$$

where y_i is the true label, p_i is the predicted probability for class 1, and N is the total number of samples. BCE Loss effectively penalizes the model based on the confidence of its incorrect predictions. However, it may struggle with imbalanced datasets, where the model may bias predictions towards the majority class. To address this, Focal Loss was introduced as an extension of BCE Loss, which down-weights the loss for easy-to-classify examples and focuses on hard-to-classify instances. Focal Loss for a BCE Loss is defined as [33]:

$$\text{Focal Loss} = \alpha(1-p_i)^{\gamma} * \text{BCE Loss}, \tag{2.10}$$

where α_c is a class-dependent weight factor, addressing the class imbalance, and γ is the focusing parameter. By adjusting γ, the Focal Loss provides a hyperparameter for focusing on low-confidence predictions in case of difficult examples, leading to better performance on rare or hard-to-detect classes [33].

2.4. 3D Object Detection & Tracking (LiDAR-based)

While 2D object detection has advanced significantly with the upcoming of deep learning, it is inherently limited to capture depth information which hinders an accurate localization of objects within 3D environments as in autonomous driving. In contrast,

sensors like LiDAR provide depth information in the form of 3D point clouds, enhancing 3D perception systems. Understanding their spatial information in terms of object classification and tracking can be realized by 3D deep learning models.

Given a 3D data source, 3D object detection aims to locate and classify all existing objects of interest (Figure 2.2). A detection $d \in \mathbb{R}^8$ consists of an object class c and a 3D bounding box represented with center coordinates (c_x, c_y, c_z), box width w, height h, and length l, and the object's heading angle θ.

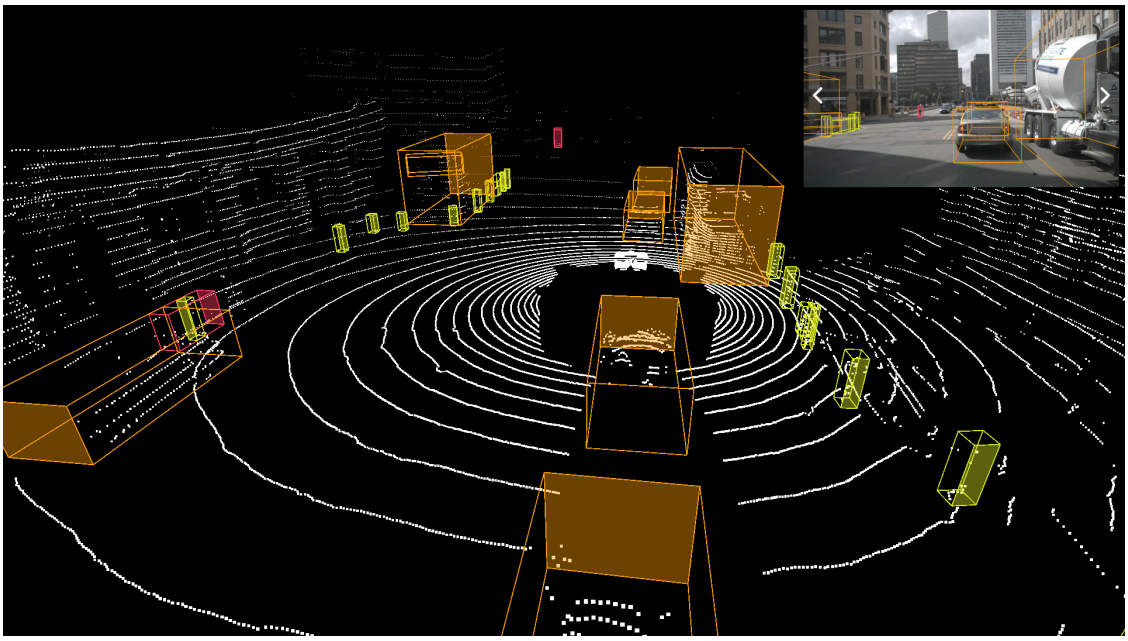


Figure 2.2.: LiDAR point cloud and frontal image view (top right) from nuScenes' Scene-0100, with bounding boxes: orange for vehicles, yellow for traffic cone, and red for cyclists [14].

In the following, object detection is further elaborated based on our used base-detector CenterPoint [60].

2.4.1. CenterPoint - 3D Detector

CenterPoint [60] is a state-of-the-art model for object detection and tracking that identifies objects based on their center points. It achieved leading performance on the nuScenes benchmark for both 3D detection and tracking and ranked first among all LiDAR-only models on the WOD.

CenterPoint was introduced with two different versions of encoding (VoxelNet [63] and PointPillars [29]) and two different stages of feature extraction. In this work, we reference the one-stage CenterPoint model with VoxelNet encoding (Figure 2.3), as the two-stage model and PointPillars versions did not improve performance on the

nuScenes benchmark.

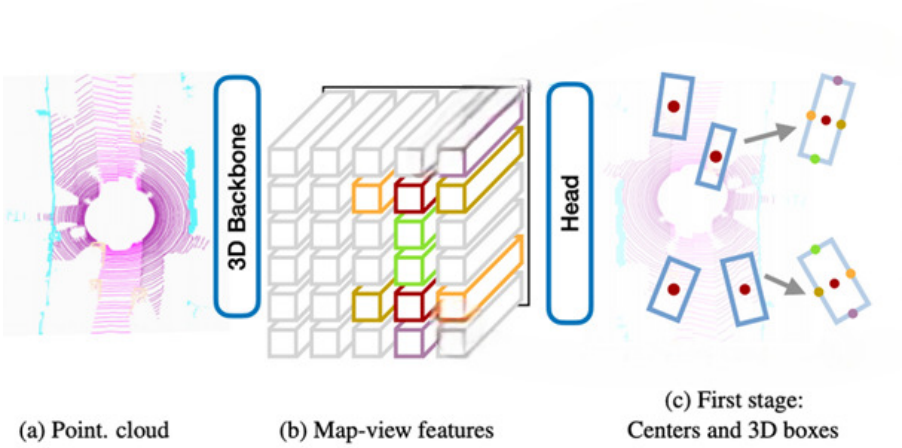


Figure 2.3.: Overview of CenterPoint's one-stage framework, adapted from [60]. Starting with a LiDAR point cloud as input (a), the 3D backbone VoxelNet extracts spatial features in bird's-eye view (b). The detection head then identifies object centers and uses their features to predict complete 3D bounding boxes (c).

To address the irregularity of point clouds and to leverage well-developed sparse convolution networks, CenterPoint employs voxelization as a preprocessing step. This process transforms raw 3D point cloud data into structured voxel grids, discretizing the continuous 3D space into uniform, fixed-sized grid cells (voxels) to enable efficient spatial feature extraction.

The extracted, spatial features are processed in bird's-eye view, which simplifies the detection task by reducing one dimensionality and mitigating frontal-view occlusions. Using this feature space, the model employs classification and regression heads based on 2D convolutional neural network (CNN) architecture [27]. The model's classification head predicts the centers of objects within such map-views by generating a heatmap targeting a 2D Gaussian, where the intensity of the peak indicates detection confidence. For each detected center, regression heads predict missing properties, such as the center height, scale and orientation of the bounding boxes. Additionally, CenterPoint assigns detection scores to object detections, representing the model's confidence that a detected object belongs to a specific class.

2.5. 3D Object Tracking (LiDAR-based)

Object tracking strongly relates to object detection due to the traditional tracking-by-detection paradigm [4, 3]. Given a sequence of source data, including detected objects within their bounding boxes, object tracking aims to estimate the trajectories of each

object along each time step (Figure 2.4). Instead of a generic class label, each object has a unique object id, tracked from frame to frame. Various matching algorithms can be applied to achieve a temporal association of different detections based on their features.

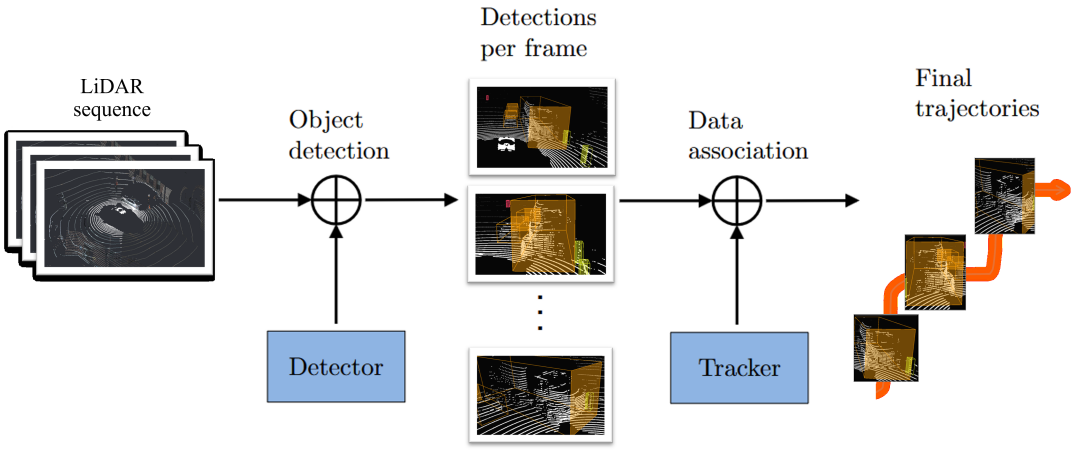


Figure 2.4.: Tracking-by-detection paradigm for 3D point clouds: Given a sequence of LiDAR point clouds, an object detector predicts bounding boxes for objects in each frame. These detections are then associated across frames by an object tracker, forming continuous trajectories for each object over time.

2.5.1. ImmortalTracker - 3D Tracker

In this section, we present an example of a 3D object tracker: ImmortalTracker [55], which we used in our work. Built on CenterPoint as its base detector, ImmortalTracker stands out for its ability to significantly reduce identity switches (IDS) when evaluated on nuScenes and WOD. Identity switches, where an object is assigned to the wrong track and receives a mismatched ID, are critical to minimize for accurate data annotation and reliable tracking systems.

To associate detections across different frames, Hungarian matching [28] is employed based on 3D-IoU [57]. This approach minimizes a cost matrix by optimizing assignments based on spatial distance and feature similarity. Unlike traditional methods that terminate tracks when objects are not detected for a few frames, ImmortalTracker always maintains tracks for all objects (Figure 2.5), avoiding identity switches caused by premature track termination. For comparison, in evaluation on the nuScenes test dataset, ImmortalTracker had only 365 IDS, while CenterPoint's tracking implementation had 760 IDS.

New tracks are initiated if detections are not assigned to existing tracks, and tracks without a match in the current frame are maintained using a vanilla 3D Kalman filter [24]. The Kalman filter serves as a motion model to predict object's next state based on its

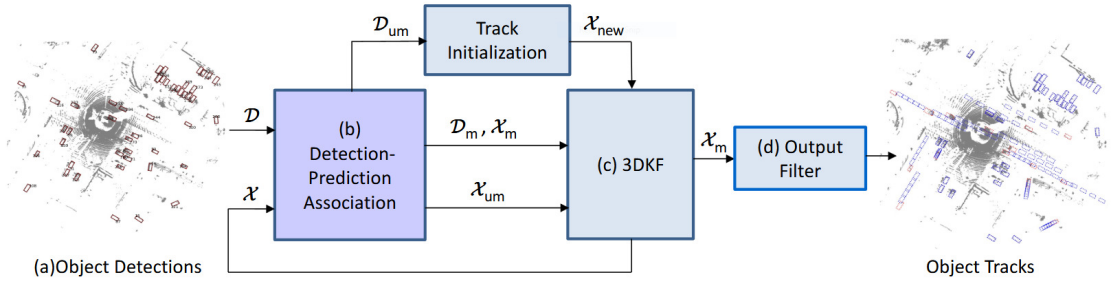


Figure 2.5.: **(a)** Start with a pre-trained 3D object detector to generate 3D bounding box detections (D) from the input point clouds. **(b)** For data association, calculate 3D-IoU between detected bounding boxes of different frames. Hungarian matching is then performed based on these similarity scores. Matched tracks (X_m) are updated using their corresponding detections (D_m), unmatched tracks (X_{um}) are updated with predicted states by a 3D Kalman Filter (3DKF) **(c)**, and unmatched detections (D_{um}) are initialized as new tracks (X_{new}). **(d)** Only active tracks that have exited their initialization stage and were successfully matched in the current frame are included in the outputs. Figure is from [55].

previous visible state under the assumption of constant velocity. To avoid the creation of false positives, the filter's predicted states contribute to the maintained track only if the object reappears as an actual detection.

2.6. Anomaly Detection

Anomaly detection identifies out-of-distribution, unknown or unexpected data samples, enhancing robustness and safety in real-world applications [59]. Unlike traditional object detection, which focuses on predefined classes, our anomaly detection approach prioritizes the identification of incorrect 3D object detections within a track environment. In this section, we provide a brief overview of prominent research in anomaly detection. This overview does not include our specific objective because, to the best of our knowledge, our work is the first study on anomaly detection in object detection.

Common application areas include industrial quality control [17, 18], surveillance [59, 47], and cyber-security [50, 13]. In cyber-security, anomaly detection typically relies on features of network activity to identify threats, while in surveillance, it often involves analyzing video data to detect unusual behavior. For industrial quality control, anomaly detection uses image or point cloud data to identify defects in items or products.

Recently, 3D anomaly detection in point clouds has gained attention, particularly in industrial quality control [34, 45, 23]. These methods often leverage multi-modal models that incorporate additional features, such as image data, to enhance detection

accuracy (Figure 2.6).

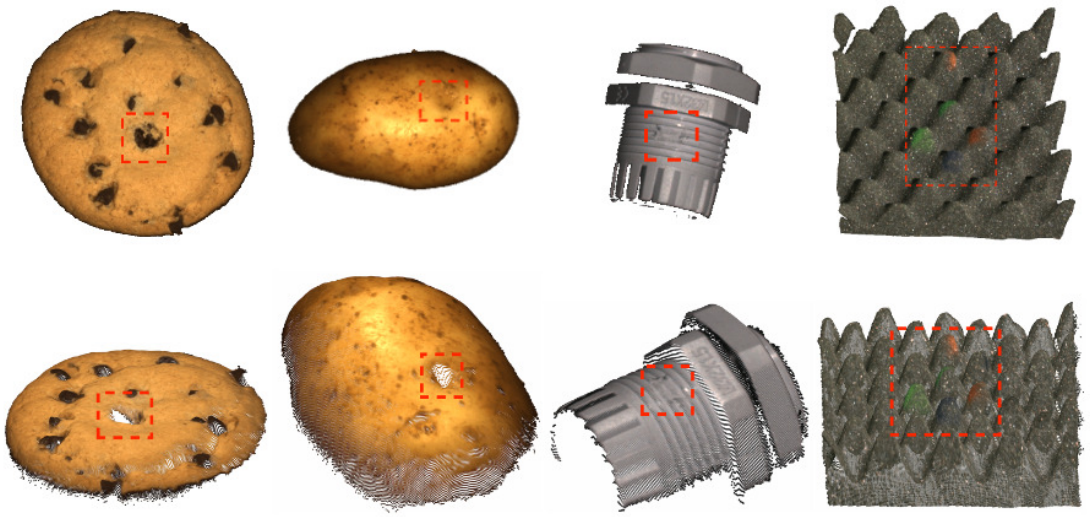


Figure 2.6.: Anomalies are marked in red. The two leftmost objects have anomalies in their shape which can be detected as deformations in 3D point clouds. The two rightmost objects have anomalies in their texture, e.g. colored foam, which can be detected within RGB images. Figure is adapted from [23].

2.7. 3D Data Labeling

Manual labeling of objects in 3D point clouds is a crucial step in the development of accurate datasets. Real world datasets like nuScenes and the Waymo Open Dataset rely heavily on human annotators to assign semantic labels to the points captured by LiDAR sensors. The purpose of this manual labeling is to provide accurate and consistent ground truth data, enabling algorithms to learn to recognize and classify various objects within complex, dynamic environments [5].

This process typically involves visualizing the 3D point cloud data using specialized software tools, where annotators interactively select and label objects based on their shape, size, and position in the scene. This requires annotators to have a solid understanding of both the sensor data and labeling conventions, dealing with challenges such as occlusions, varying object shapes, and incomplete point clouds [5].

Automated labeling models exist that achieve human-level results or even surpass them in some cases [12], though they require extensive training on labeled data. In contrast, semi-automated labeling models additionally rely on human interaction. For instance, "Click, Crop & Detect" [25] is a method involving a human clicking on the center of an object in a 3D point cloud, cropping the area around the click, and then using a 3D object detector to accurately identify the object (Figure 2.7).

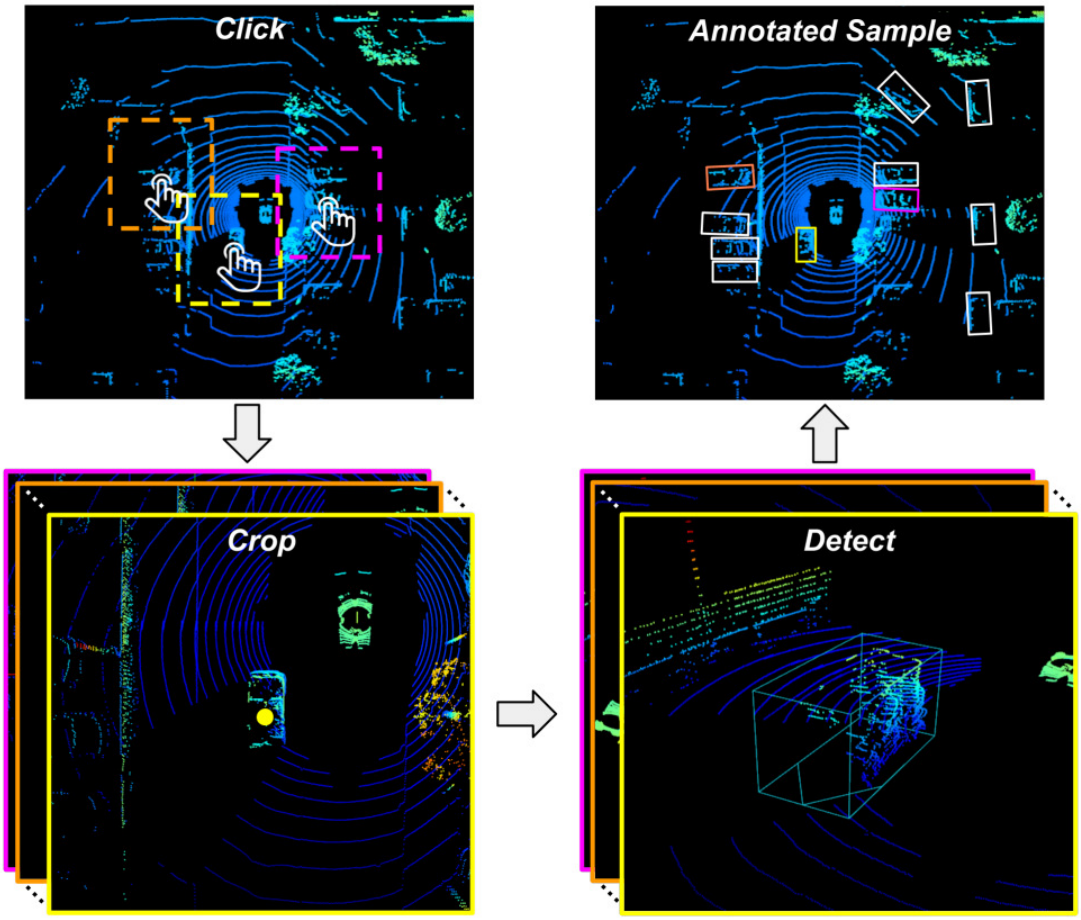


Figure 2.7.: Pipeline of "Click, Crop & Detect": Click: Annotators click on the center of objects of interest within a 3D point cloud. Crop: The point cloud is cropped around the annotator's selected center. Detect: The cropped point clouds are processed by a 3D object detector, resulting in precise 3D bounding boxes within the whole point cloud. Figure is from [25]

3. Related Work

After introducing the background knowledge of this work, this chapter delves deeper into research closely related to our topic. We begin with an overview of various studies on semi-supervised learning (see 3.1), which we considered for our work and may also offer valuable insights for future work, leveraging even more performance potential. Our primary focus lies in developing models for automated data labeling, drawing inspiration from recent advancements in this domain, as outlined in Section 3.2.

3.1. Semi-Supervised Learning in 3D Object Detection

Semi-supervised learning (SSL) is a machine learning paradigm where a model is trained using a small set of human-annotated data combined with a large set of unlabeled data. Since human-annotated data is often limited and costly to obtain, SSL aims to leverage unlabeled data more effectively to improve model performance.

SSL approaches can be broadly categorized into two main paradigms [37]:

1. Pseudo-labeling: Utilizing model-generated predictions on unlabeled data as pseudo ground-truth labels for further training.

2. Consistency Regularization: Enforcing consistent predictions across augmented versions of the same input data.

While these paradigms are often combined in practice, extensive research has explored each approach independently to address unique challenges and optimize performance. In this work, we do not investigate a new learning strategy for SSL, and just use standard self-training as a test case to undermine our concept and to showcase its potential.

3.1.1. Data Augmentation

Data augmentation involves applying transformations to data, and possibly its annotations, to create new training samples. For example, in a 3D object detection task using point clouds, we could apply random rotations and translations to a set of points representing a car while ensuring that its bounding box annotation is updated accordingly [64].

Data augmentation can enrich smaller datasets with new examples and prevent overfitting. In many projects, including ours, basic point cloud augmentation has been used, such as the dropping, shifting, rotating and flipping a set of points [60].

There is research on more application-specific 3D augmentation including different data modalities, such as images, or synthetic data to exploit different data domains [64]. However, we focus our work on a single dataset and 3D point clouds.

An interesting advanced extension of this approach is training a model to learn such augmentation noise, enabling the generation of highly realistic samples. One example is *Adversarial Deformation* [64], an advanced 3D augmentation method where a separate model generates new data samples through competitive training with a classifier. For instance, *3D-VField* [30] focuses on generating out-of-domain samples, particularly rare and damaged cars, by smoothly deforming instance shapes until they appear realistic to the classifier.

However, such methods typically require expensive training to produce accurate and realistic samples. Further, we decided to avoid an adversarial framework, as the priority of our work is producing an accurate detector for false positives, ideally supported by data augmentation, rather than generating new, realistic false positives.

3.1.2. Pseudo-labelling

A pseudo label is an artificial label assigned to unlabeled data based on predictions from a trained or partially trained model, rather than human annotation, enabling its use as labeled data during training.

Recent efforts [53, 37, 62, 54, 7] have focused on developing sophisticated student-teacher frameworks for 3D semi-supervised learning, originally introduced for image classification [49]. In these frameworks, teacher models guide student models by comparing predictions on the same sample, with stronger data augmentations applied to the student's input. Building on this foundation, approaches have been adapted for 3D object detection, employing techniques such as consistency losses between teacher and student predictions to improve detection performance with limited labeled data [53, 37, 62].

Some methods extend this idea further. For instance, one approach enhances the teacher model with spatiotemporal reasoning using a graph neural network, iteratively refining the student model with pseudo labels generated from a large set of unlabeled scenes [54]. Another method addresses the challenge of low-quality pseudo labels in distant regions by incorporating dense imagery data into the teacher model, achieving over 90% of fully supervised performance with only 5% labeled data [7].

Despite their successes, these methods remain highly sensitive to teacher model performance, which may amplify detection biases [8]. This is why we favor a simpler yet effective method: self-training via pseudo labels. In this approach, a model is trained on its own predictions for unlabeled data. Methods that use self-training are less focused on optimizing the training process itself, as seen in the teacher-student framework. Instead, they prioritize reducing the reliance on labeled data.

For example, frameworks leveraging unlabeled multi-view image data, such as stereo or video, introduce object-wise consistency losses to provide external 3D supervision, effectively reducing the dependency on extensive labeled datasets [31]. Similarly,

pretraining methods that utilize geometry-aware contrastive learning and clustering techniques have demonstrated significant improvements in performance, even when trained on subsets of labeled datasets [32]. These approaches highlight promising directions for advancing 3D semi-supervised learning by prioritizing data efficiency and minimizing reliance on complex teacher-student interactions.

3.2. Automated Data Labeling Models

Tools for (semi-)automated labeling are often employed in human-in-the-loop data processing pipelines, reducing manual effort and enhancing model performance [58]. Commercial tools such as Scale AI [1], used for annotating datasets like nuScenes [22], offer extensive automation via machine learning pipelines, quality control mechanisms, and scalable workflows for large datasets. Free-to-use alternatives are for example CVAT [9], offering semi-automated labeling by box interpolation, or Xtreme1 [15] with the option of custom automation by integrating own machine learning models.

Published research on models for automated 3D data labeling in autonomous driving is limited to this date [61, 12, 35]. One of the first published methods leveraged sparse LiDAR point clouds and 2D detections from pre-trained off-the-shelf 2D detectors to predict 3D shapes and their bounding box coordinates [61]. This approach refined predictions iteratively using a differentiable renderer for signed distance fields, pre-trained on synthetic data. A curriculum learning strategy further enhanced this process by gradually improving label accuracy with retraining on increasingly challenging samples.

Building on these foundations, more recent approaches [12, 35] have shifted towards track-centric learning, inspired by human annotation practices. Instead of treating objects in isolation within individual frames, these methods analyze object tracks across multiple frames. This track-centric approach benefits from offline settings, allowing for bidirectional analysis of object movements both forward and backward in time, leading to improved object localization and robust tracking even in cases of temporary occlusion. By incorporating both temporal and spatial information, these approaches achieve state-of-the-art results on challenging 3D object detection benchmarks.

In these frameworks, consecutive frames containing tracks and associated point clouds are processed through refinement models designed to improve the accuracy of predictions. These models refine false detections, adjust localization, or assign low confidence scores to incorrect predictions, thereby enhancing overall accuracy. A specialized metric, Track Intersection over Union (TIoU), has been introduced to evaluate track-level performance by measuring the overlap between predicted and ground truth tracks over their union [12].

While significant progress has been made, automated labeling for 3D data in autonomous driving remains an active research area with notable limitations in current approaches [12, 35]:

1. **High computational cost:** Current methods rely on complex models that re-

quire significant computational resources, which may pose a barrier for smaller organizations due to high training and deployment costs.

2. **Data dependency:** Heavy reliance on large labeled datasets raises the question of whether similar performance can be achieved with less data, particularly in domains like autonomous driving where annotations are expensive [48].

3. **Underrepresented classes:** Existing methods, developed on datasets with limited class diversity (e.g., Waymo Open Dataset [48]), struggle to address rare or safety-critical categories due to the lack of a tail distribution.

Addressing these challenges requires lightweight, scalable, and semi-supervised solutions capable of balancing efficiency, accuracy, and robustness across diverse settings.

4. Methodology

Track refinement approaches ([35], [12]) have proven significant potential in leveraging tracking data structures for approving 3D detections in an offline manner. Inspired by their solution approaches, we developed a novel approach and benchmark focused solely on detecting false object detections (anomalies) within such tracks rather than refining locations of bounding boxes.

In this work, we define anomalies and nomalies as follows:

- **Anomaly**: The center distance between CenterPoint's predicted bounding box and the ground truth bounding box exceeds or is equal to 4 meters.
- **Nomaly**: The center distance to the ground truth bounding box is less than 4 meters.

Anomalies correspond to false positive predictions, which we aim to filter (Positives for the anomaly detector). Conversely, nomalies represent valid detections that should be retained (Negatives for the anomaly detector). Our solution approach (Figure 4.1) is universally applicable to any object detector, tracker and refinement module.

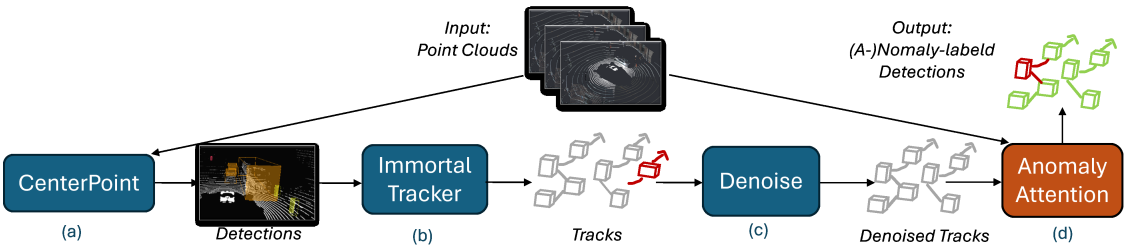


Figure 4.1.: Framework overview: Data preprocessing steps are (a), (b), and (c). The detection model **(a)**, CenterPoint [60], predicts 3D bounding boxes from a point cloud input. The tracking model **(b)**, ImmortalTracker [55], generates tracks based on these detections, which are denoised **(c)** based on their lengths. Finally, our anomaly detector **(d)** identifies anomalous detections within these tracks, leveraging track features and their associated point clouds.

4.1. Data Preprocessing

Our preprocessing consist of three modular steps:

1. Object detection via CenterPoint [60]
2. Track generation via ImmortalTracker [55]
3. Track denoising by length filtering

Similarly to other implementations ([35], [12]), we use CenterPoint [60] (step (a) in Figure 4.1) as our base detector to generate detections from CenterPoint's detections on LiDAR point clouds (Figure 4.2). In the semi-supervised test case, CenterPoint undergoes a retraining step using self-training via pseudo-labels. Next, object detections are labeled as either an anomaly or a nomaly.

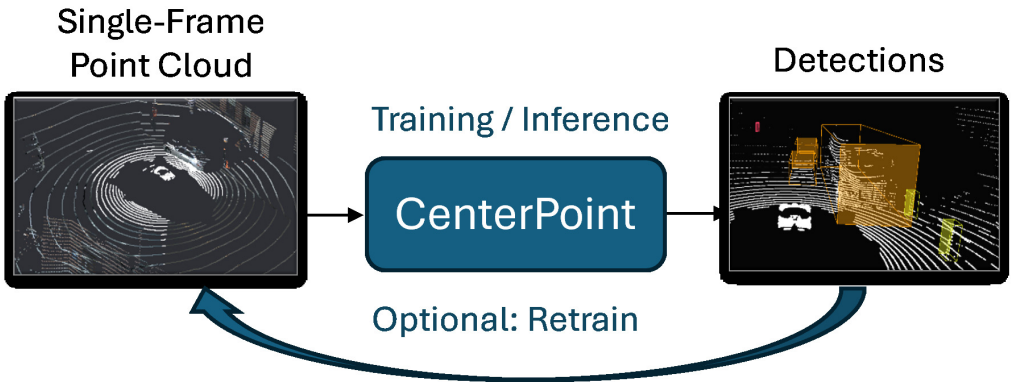


Figure 4.2.: CenterPoint predicts 3D bounding boxes from LiDAR point clouds and compares them with manually labeled ground truth during training. Optionally, CenterPoint can generate pseudo-labels for unlabeled data and undergo retraining based on its detections.

CenterPoint's single-frame detections are enhanced by generating tracks with temporal context using ImmortalTracker [55] (step (b) in Figure 4.1). Tracks group detections across multiple LiDAR frames, transforming them into a sequence-based representation (Figure 4.3). For simplicity, we do not apply any major changes to CenterPoint and ImmortalTracker. However, as done in CTRL [12] for ImmortalTracker, we allow tracks to be initialized immediately after a single detection instead of requiring multiple consecutive detections. To balance this increased track creation, we raised the confidence threshold from 0.01 to 0.1.

To reduce noise introduced by ImmortalTracker, tracks shorter than 5 detections are removed (step (c) in Figure 4.1), which improves overall precision (Figure 5.1, see Section 5.1). In the following, we describe our data structure and tested data augmentation methods.

4.1.1. Data Structure

For anomaly detection, we utilize two distinct data modalities as input (Figure 4.4):

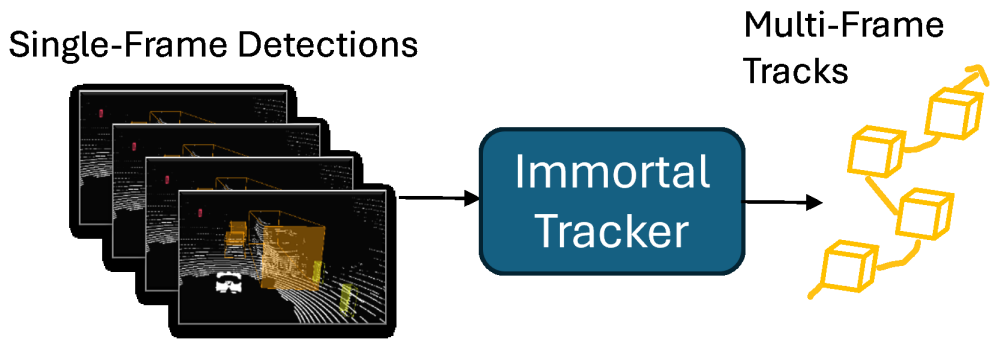


Figure 4.3.: Track generation by ImmortalTracker: transforming single-frame detections into multi-frame tracks for temporal context.

- LiDAR point cloud sequence (41 frames)
- Tracks of object detections within such point cloud sequence

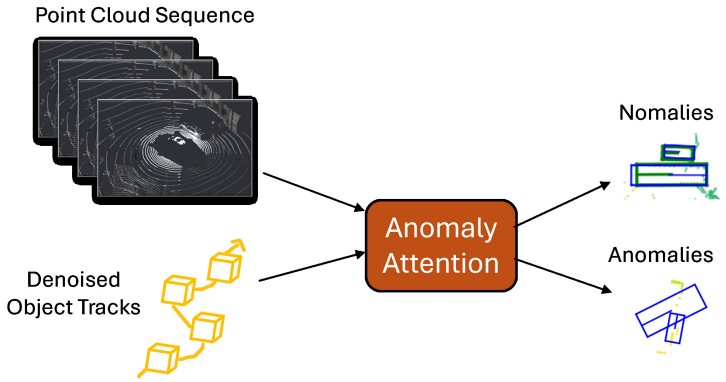


Figure 4.4.: Using object tracks and their corresponding sequences of point clouds, our anomaly detector identifies anomalous detections within the tracks.

Each frame's point cloud is represented as a matrix of shape $(N, 5)$, where:

- N denotes the downsampled number of points (approximately 16k - as in the original CenterPoint implementation).
- Each row corresponds to a single LiDAR point and contains the following five features:
  - **x-coordinate** ($\mathbb{R}$)
  - **y-coordinate** ($\mathbb{R}$)
  - **z-coordinate** ($\mathbb{R}$)
  - **Intensity** ($\mathbb{R}$): The reflection strength of the LiDAR beam at the point.

– **Timestamp** ($\mathbb{R}$)

A track $t \in \mathbb{R}^{i \times 9}$ consists of i detections, all sharing the same tracking ID generated by ImmortalTracker. From the detected 3D bounding boxes by CenterPoint, we extract the following attributes as features for each detection $d \in \mathbb{R}^9$ within a track:

- **Translation** ($\mathbb{R}^3$): The 3D position of the bounding box center in space, represented by (x, y, z) coordinates.
- **Rotation** ($\mathbb{R}^1$): The yaw angle of the bounding box, describing its orientation in the horizontal plane.
- **Scale** ($\mathbb{R}^3$): The size of the bounding box along the (x, y, z) axes.
- **Number of LiDAR Points** ($\mathbb{R}^1$): The number of LiDAR points enclosed within the bounding box.
- **Class Number** ($\mathbb{Z}^1$): An integer indicating the object class (e.g., car, pedestrian).

To process tracks from the same sequence of frames together in a batch (Figure 4.5), we apply padding to ensure uniformity: all tracks have the same lengths (41 detections), and all frame sequences contain the same number of tracks (maximum of all sequences, around 2300). This ensures that all tracks in a batch share the same sequence of point clouds. For computational efficiency, batches composed entirely of padding are skipped during training. Instead of padding with zeros or ones, we use a distinctive placeholder of -500, which is masked out in the loss calculation. This value is clearly outside any realistic range, as the only features that can have negative values are rotation (minimum -180°) and LiDAR point coordinates (minimum -70 meters).

4.1.2. Data Augmentation

To increase dataset size and create unseen samples, we applied data augmentation to the ground truth data. A simple yet effective strategy we used is carefully adding random noise to ground truth data to generate new samples. We ensured that the applied noise remained within specific thresholds to guarantee that modified samples either stayed true positives or became anomalies (Figure 4.6).

During evaluation, predicted bounding boxes are classified as anomaly based on their center distance from their corresponding ground truth boxes in a bird's-eye view. Therefore, we primarily augmented the x and/or y translation values of each 3D bounding box (Algorithm 1) by shifting its center to a distance within the range of $(4.5, 9]$ meters. This ensures that the shift introduces an anomaly, as for a true positive, the maximum allowable distance is 4 meters. We also apply random factor in range $[0.5, 1.5]$ to all other detection features (4.1.1).

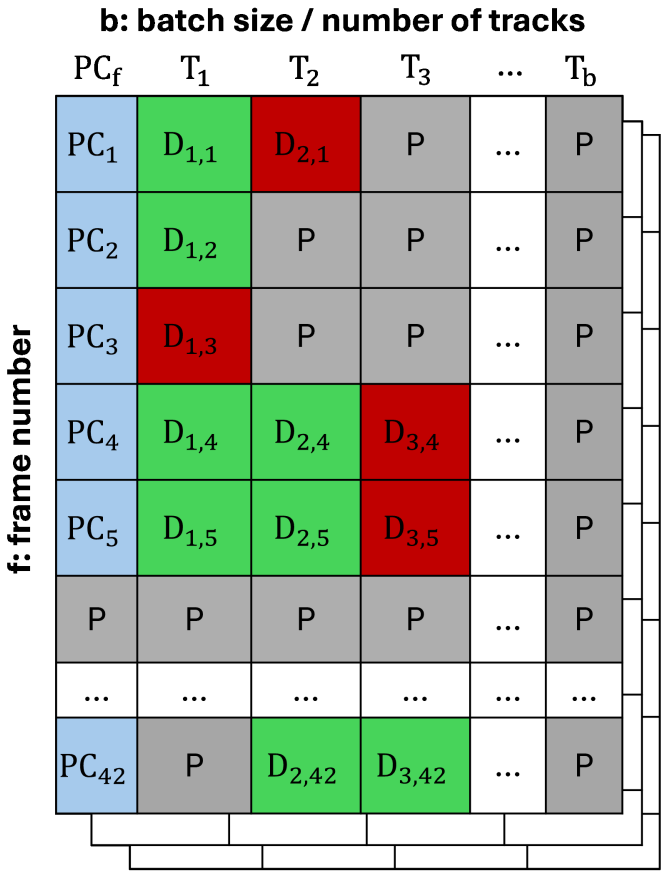


Figure 4.5.: Batch representation: Each batch has for each frame (f) (41 per scequence) point cloud features and a number of b tracks, where b is the batch size. Each track can have detection features (D_b, f) corresponding to its object in each frame. Detections are labeled as anomalies (red) or nomalies (green). Point cloud features are padded (P) if no objects are detected in a frame (also padded) across all tracks in the batch. Entire tracks may also be padded if they belong to the last batch of a shorter sequence than the maximum sequence length (T_b).

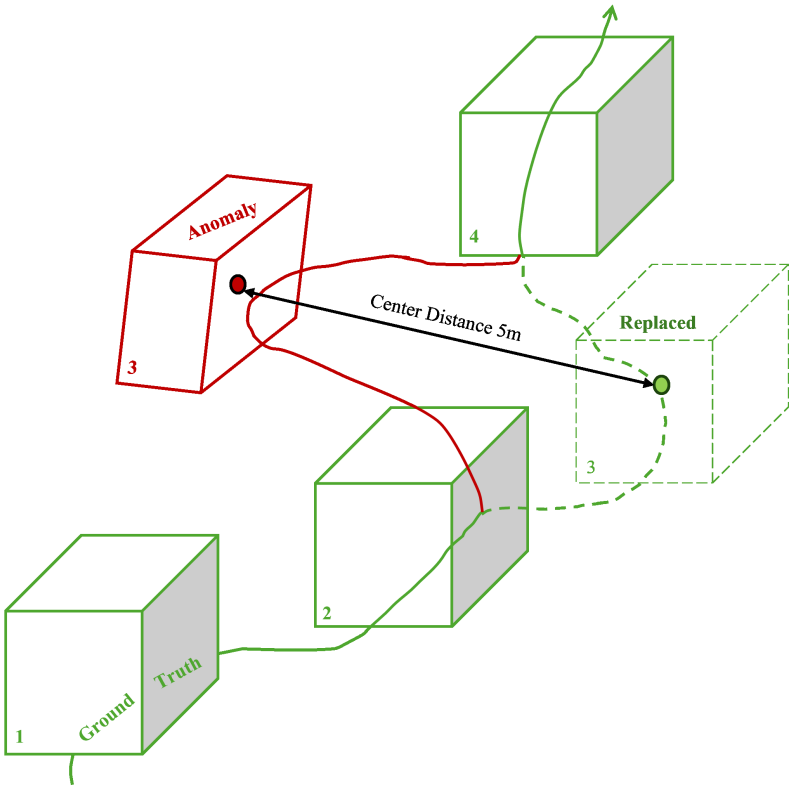


Figure 4.6.: Augmentation concept: Given a nuScenes ground truth track with four detections, we apply an 80% chance of augmenting a ground truth detection (bounding box 3) into an anomaly, altering the overall characteristics of the track.

Algorithm 1 Translation Noise Augmentation for Anomaly Creation

Input: Bounding boxes' translation vector $\mathbf{t} = (x, y, z)$, augmentation probability p
Output: Augmented translation vector $\mathbf{t}'$

if **RandomProbability()** $> p$ **then**
  return $\mathbf{t}$ ▷ Skip augmentation
end if
$\mathbf{n} \leftarrow$ **RandomNoiseVector**$(\mathbf{t})$ ▷ Create noise vector
$\mathbf{n} \leftarrow \mathbf{n}/||\mathbf{n}||$ ▷ Normalize so $||\mathbf{n}|| = 1$ and $\mathbf{d} \in (-1, 1)$ for all $\mathbf{d} \in n$
Select $k \leftarrow$ **RandomDimensions**$(x, y, both)$ ▷ Select one or both dimensions
if $k =$ **both then** ▷ Scale n so that $\|\mathbf{n}\| \in (4.5, 9]$
  Scale $\mathbf{n}$ in dimensions $\mathbf{k}'$ by random value in range $[4, 5]$
else
  Scale $\mathbf{n}$ in dimension $\mathbf{k}'$ by random value in range $[5.5, 8]$
end if
$\mathbf{t}' \leftarrow \mathbf{t} + \mathbf{n}$ ▷ Apply noise to translation
return $\mathbf{t}'$

Using Optuna [2], we tested different hyperparameter settings weighing augmented samples down to half as much as real samples.

4.2. Training Framework

We train and test our base detector and anomaly detector on the same training and validation datasets. During the development of our anomaly detector, we used 15% of the training data to monitor a binary cross-entropy loss (see Section 2.9) as the validation loss. The loss function used for training is detailed in the following section. For both loss functions, padding values are masked out during loss calculation.

As the goal of our work is to minimize human labeling effort, we also trained our model on smaller seed datasets to demonstrate its effectiveness with limited labeled data. Detailed experimental settings are discussed in our evaluation Chapter 5, but we provide an overview of the general training procedure in the following sections.

4.2.1. Loss Function

Predictions are labeled as anomalies if their center distance to the ground truth bounding box is equal to or greater than 4 meters, as they are considered false positives across all classes and distance thresholds in the nuScenes' benchmark (see Section 2.2.2).

To address the class imbalance between anomalies and nomalies, we use a focal binary cross-entropy loss (see Section 2.10) with a standard gamma value of 2 and balancing alpha weights, calculated by balanced heuristic [26] [46]:

$$w_c = \frac{N}{C \times n_c}, \tag{4.1}$$

where N is the total number of samples, C is the total number of classes, n_c is the number of samples of class c. This heuristic ensures minority classes receive proportionally higher weights. To avoid vanishing gradients caused by the sigmoid activation function, we employ PyTorch's binary cross-entropy with logits loss [39] (see Section 2.3). The loss is given by the average of the per-sample binary cross-entropy terms:

$$\ell(x, y) = \frac{1}{N} \sum_{n=1}^{N} \Big[-w_n \Big(y_n \log\big(\sigma(x_n)\big) + \big(1 - y_n\big) \log\big(1 - \sigma(x_n)\big) \Big) \Big], \tag{4.2}$$

where x_n are the logits, $y_n \in \{0, 1\}$, $\sigma(\cdot)$ is the Sigmoid function. This formulation combines the Sigmoid-based classification and loss calculation into a single operation, improving numerical stability.

Additionally, we scale the alpha weights (see Section 2.10) for nomalies by 50% based on their quality relative to nuScenes distance thresholds of [0.5, 1, 2, 4] meters (Figure 4.7). This weight scaling methods values true positive samples more if they are closer to ground truth's center than other samples. For example, a nomaly prediction receives a 250% higher weight if its predicted center aligns with the ground truth across all distance thresholds rather than just the 4-meter threshold (no scaling).

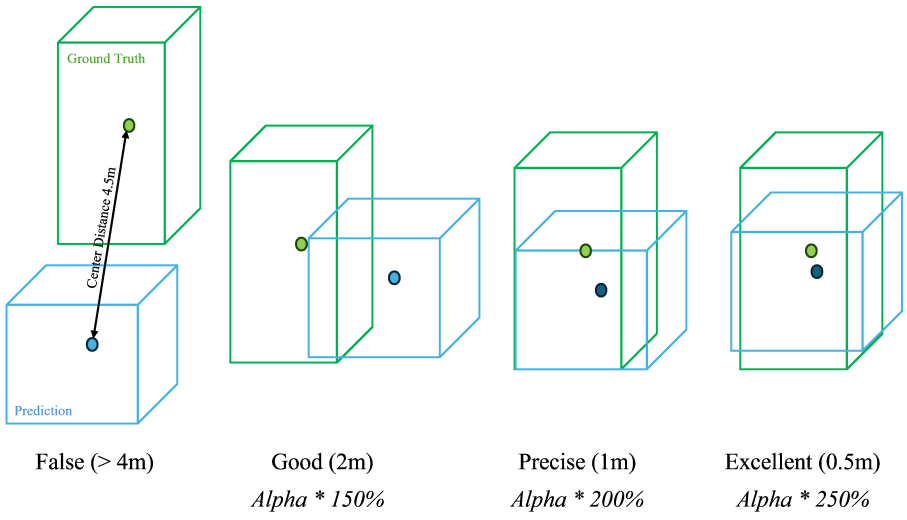


Figure 4.7.: Prediction quality based on center distance between predicted (blue) and ground truth (green) bounding boxes: NuScenes' detection benchmark has four distance thresholds [0.5, 1, 2, 4] meters. A true positive within a 0.5-meter distance is also considered a true positive for all higher thresholds. To account for this, we scale the alpha weights for all samples in our loss function by 50% increments based on the distance metric. Predictions exceeding the 4-meter threshold are considered as false positives and labeled as anomalies.

4.2.2. Semi-Supervised Training - Base Detector

To demonstrate the potential of our method in minimizing human labeling effort, we employ semi-supervised training. Specifically, we use a small seed dataset consisting of 5% of the full training sequences while treating the remaining sequences as unlabeled data. We ensure that the class distribution in the seed dataset approximately matches that of the full dataset.

Initially, the base detector is trained on the seed dataset. It then generates pseudo-labels for the unlabeled data, after which it is retrained using both the seed dataset and the pseudo-labeled data (Figure 4.8). To address the high number of false positives caused by low confidence thresholds aimed at maximizing recall, we apply our anomaly detector to filter incorrect detections after inference on the validation dataset. Importantly, the anomaly detector itself is trained exclusively on the predictions of the base detector from the seed dataset. Furthermore, our anomaly detector could also be utilized during the retraining process of the base model by filtering pseudo-labels to exclude anomalies. However, this approach was not further explored in this thesis.

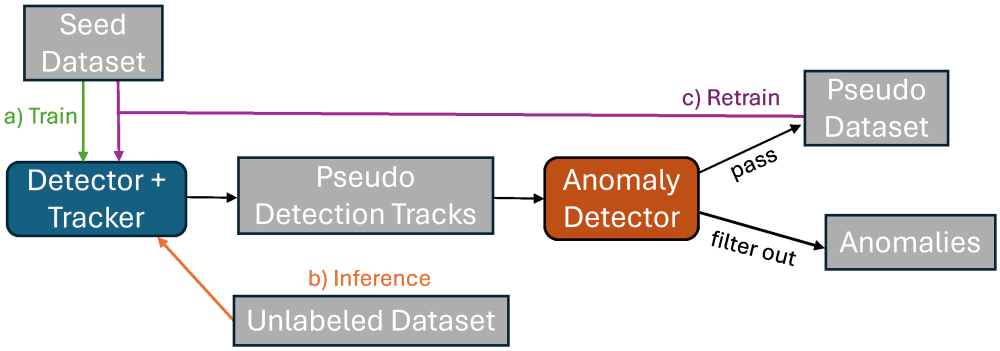


Figure 4.8.: Semi-Supervised framework: **a) Train:** An object detector and an anomaly detector are initially trained on a small, human-labeled seed dataset. **b) Inference:** The trained object detector generates pseudo labels on an unlabeled dataset. These pseudo labels are organized into tracks by a tracker and processed by the anomaly detector, which filters out anomalous detections in tracks. **c) Retrain:** The remaining, non-filtered pseudo labels are combined with the original seed dataset to retrain the object detector, enhancing overall performance.

4.3. Model Architecture

Our model is a binary classifier designed to detect anomalous detections within tracks based on their trajectories across multiple point cloud frames (Figure 4.4). During early development, we explored multiple model architectures to classify anomalies in generated tracks, using only the predicted tracks and their corresponding point clouds

as input. For extracting track features, we evaluated MLP-based [19], Transformer-based [51], and LSTM-based [21] backbones; for point clouds, we experimented with PointNet [42] and VoxelNet [63]. Ultimately, our final model, *AnomalyAttention*, utilizes an LSTM for track features, a VoxelNet backbone for point-cloud features, and attention heads for feature fusion.

During preliminary tests, we also investigated batch normalization. However, these did not improve performance, likely due to the extensive padding required by our data structure. Moreover, for all layers except those in VoxelNet, we employed Xavier uniform initialization [16], which offered a slight advantage over random or Kaiming initialization [20] in early experiments. In the following sections, we outline the intermediate approaches explored during development and describe our final architecture, called *AnomalyAttention*.

4.3.1. ResNetAD

To establish a simple baseline model for rapid development and testing, we implemented a Multilayer Perceptron (MLP). The model consists of an input layer, multiple hidden layers, and an output layer. It takes a track feature tensor as input (see Section 4.1.1) and outputs a binary classification score for each detection in a track. Inspired by ResNet [19], the model includes residual connections between hidden layers. This design allows the model to learn both transformed and original features, enhancing performance in deeper networks (see Figure 4.9).

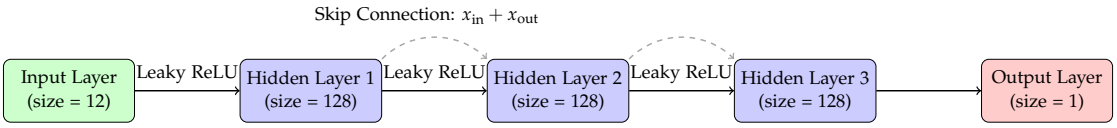


Figure 4.9.: Architecture of the ResNetAD model. Skip connections add the input of each hidden layer to its output.

4.3.2. PointNetAD

This approach integrates the former ResNetAD-based classification with a feature extraction pipeline for point clouds (Figure 4.10). It includes two main branches: the **Track Branch** (as described in Section 4.3.1) and the **Point Cloud Branch**.

To evaluate a simpler and lightweight point cloud encoder independent of CenterPoint and voxelization, we incorporated PointNet. This architecture combines an MLP, a PointNet encoder, and a joint feature extraction branch. However, this approach resulted in a performance decrease compared to the previously tested architecture. Furthermore, as PointNet was originally developed for outdoor 3D scenes, we opted to reuse our base detector's feature extractor, VoxelNet, due to its higher compatibility with our use case.

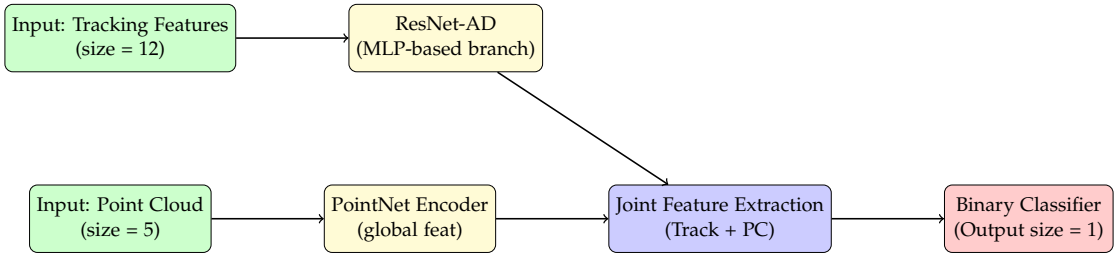


Figure 4.10.: The PointNetAD integrates a PointNet-based encoder for point cloud data and a ResNet-like MLP branch for tracking features, followed by joint feature extraction and binary classifcation by MLPs.

4.3.3. VoxelNetAD

Similarly to the former approach, using PointNet, this architecture (Figure 4.11) integrates two main branches: the **Track Branch** (as described in Section 4.3.1) and the **Voxel Branch**.

The Voxel Branch performs voxelization of the sampled input point cloud, followed by feature extraction using VoxelNet [63]. Pre-trained weights from our base detector, CenterPoint, are leveraged for the VoxelNet encoder, which outputs a feature map representing spatial context around the track.

Features from the Track and Voxel branches are concatenated, further refined through additional convolutions, and passed to a linear layer for binary classification. While this hybrid design effectively integrates spatial and track-level information, it did not result in a significant improvement in classification performance.

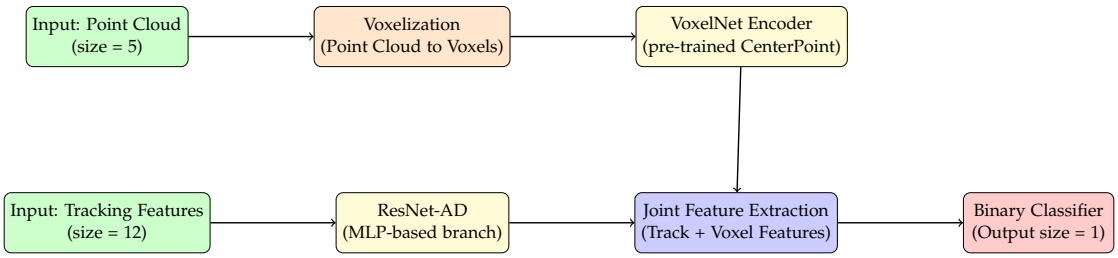


Figure 4.11.: The VoxelNetAD integrates a VoxelNet-based encoder for point cloud data and a ResNet-like MLP branch for tracking features, followed by joint feature extraction and binary classifcation by MLPs.

4.3.4. AnomalyAttention - Final Implementation

As all former implementations showed limited performance, we opt for a model architecture with higher capacity. Because our track-based data structure has temporal

context that was not exploited in previous approaches and our base detector, we include sequential reasoning using an LSTM [21] and attention layers [51]. We leverage cross-attention to provide a learnable aggregation function for fusing track and voxel features.

To capture dependencies across sequential detections, we also tested an off-the-shelf Transformers [51] [41], but training proved unstable, so we chose the off-the-shelf, bidirectional LSTM [40] for simplicity. To enhance the final classifier, we add multi-head self-attention to the classification head, effectively detecting both anomalies and nomalies. The architecture's key components (Figure 4.12) can be summarized as:

- **VoxelNet Backbone**: as described in Section 4.3.1
- **Track MLP**: Track-level features are processed independently through a multi-layer perceptron to generate compact representations for each track.
- **Cross-Attention Fusion**: Track features are fused with voxel features using a cross-attention mechanism, aligning spatial and contextual information from both modalities.
- **Temporal Modeling**: The fused features are passed through a bidirectional LSTM to capture temporal dependencies across track points.
- **Classification Head**: A multi-head self-attention layer further refines the temporal representation. Then, the features are passed through an MLP to output binary classification logits for each of the track's detections.

We applied a higher dropout rate (30%) in the early layers of the Track MLP to enhance robustness against variations in the input. In contrast, the dropout rate was reduced to 10% in the later layers to ensure smoother gradient flow. The model's final hyperparameters are based on study usingn Optuna [2] (detailed in the Appendix C).

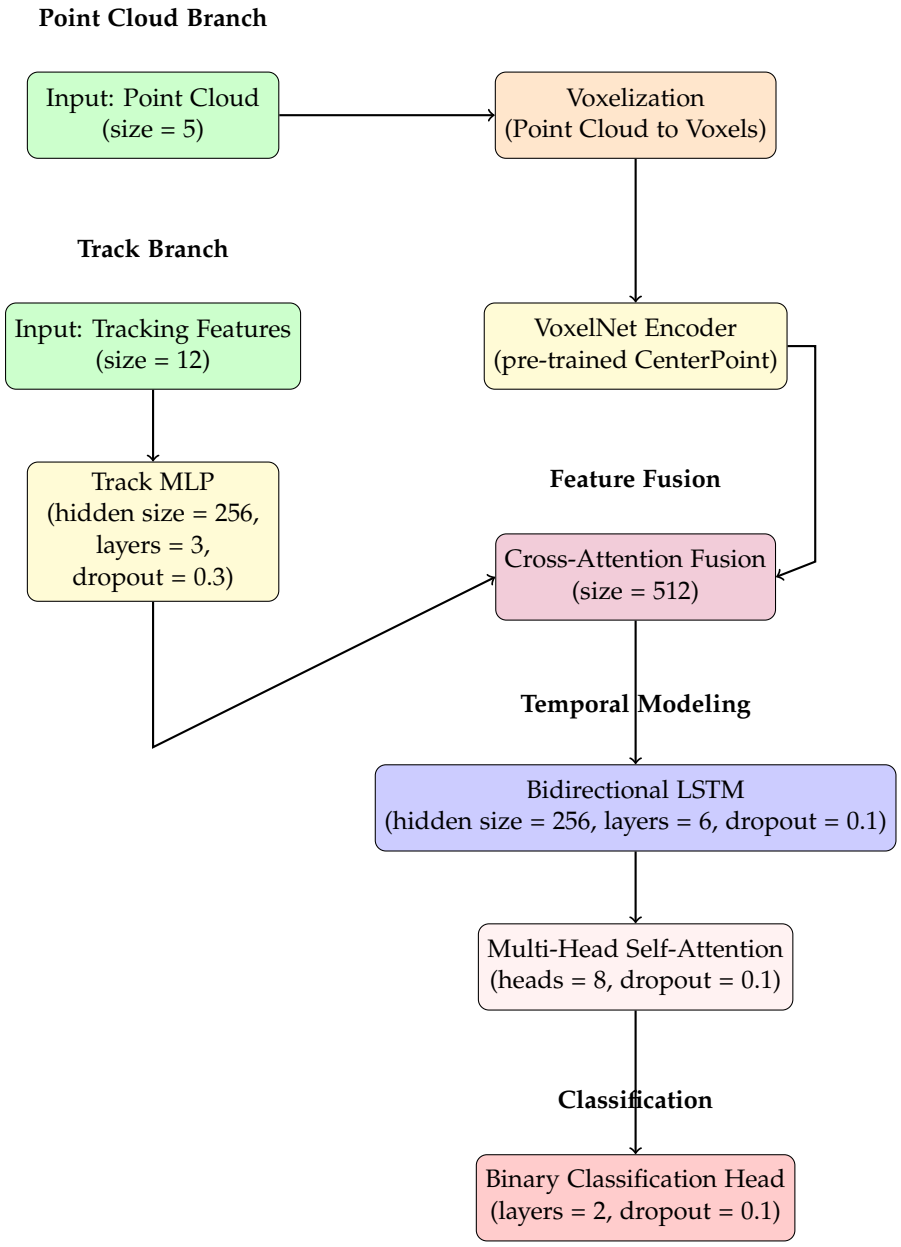


Figure 4.12.: AnomalyAttention incorporates two branches. The Point Cloud Branch extracts spatial context through a pre-trained VoxelNet encoder. The Track Branch uses an MLP to generate compact representations of track features. Cross-attention fusion aligns track and voxel features, enabling a learnable aggregation of spatial and temporal information. A bidirectional LSTM captures dependencies across sequential detections, while multi-head self-attention and classification head refine the representation, effectively identifying anomalies and nomalies.

5. Evaluation

In this chapter, we present the results for our anomaly detector: AnomalyAttention. We provide an overview of the experimental setup for all test cases, and present both quantitative and qualitative results. The performance of AnomalyAttention is assessed using binary classification metrics (Section 2.1), while the overall framework is evaluated using nuScenes object detection metrics (Section 2.2.2).

5.1. Experimental Setup

We evaluate our framework using the nuScenes dataset (Section 2.2.2), which presents a challenging tail distribution commonly seen in real-world datasets Table 5.1, emphasizing performance variations across classes. All experiments were performed using a single NVIDIA GeForce RTX 4090 GPU.

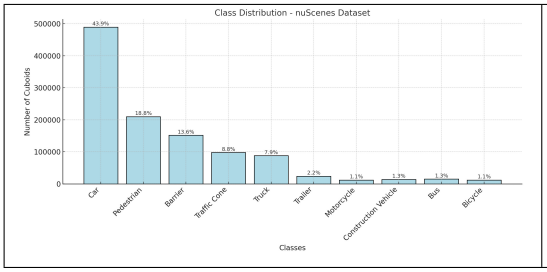
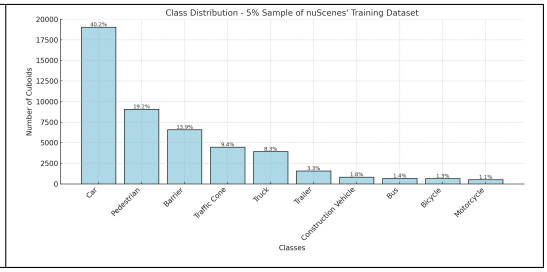
nuScenes Dataset	**5% Training Seed**
	

Table 5.1.: Class Distribution of the nuScenes dataset (left) and for 5% seed data sample of the nuScenes' training dataset (right).

The nuScenes dataset comprises 1,000 driving sequences of approximately 20 seconds each, divided into 700, 150, and 150 sequences for training, validation, and testing, respectively. These correspond to 28k, 6k, and 6k annotated frames. For our fully supervised test case, we utilize 100% of the training dataset, while the semi-supervised test case uses only 5%. Since the test dataset annotations are unavailable, we use the validation dataset for final evaluation and reserve 15% of the training dataset as a validation set for monitoring performance during development. This results in the following dataset splits:

- Training Split: x% of nuScenes' training dataset
- Validation Split: 15% of the Training Split
- Test Split: nuScenes' validation dataset, used for our final evaluation

The primary evaluation metric for nuScenes' object detection benchmark is mean Average Precision (mAP), which is calculated by averaging the per-class AP values, where each class's AP is the mean of its distance-based AP values across multiple thresholds (see Section 2.2.2). This metric inherently weighs true positives higher than false positives, as true positives contribute to both Recall and Precision (see Section 2.1). We adopt mAP as our performance metric to align with our goal of minimizing human labeling effort, which is more labor-intensive for annotating 3D objects (true positives) than for removing incorrect annotations (false positives).

Following the original implementation, CenterPoint is trained for 20 epochs and a batch size of 22 on all classes, including *car* and *pedestrian*, with a detection score threshold of 0.1. The training process required up to 20 days to complete. We exclude the majority classes, *car* and *pedestrian*, from our evaluation and from training and testing our anomaly detector, as the improvement margin for their mAP is comparably minimal across our test cases (Figure A.1, Figure B.1).

Detections are labeled as anomalies if their center distance to the ground truth exceeds 4 meters (false positives for all of nuScenes' distance thresholds) and as nomalies otherwise (Section 4). Next, tracks are generated using ImmortalTracker and filtered to exclude those with lengths below 5, reducing noise and improving mAP (Figure 5.1)."

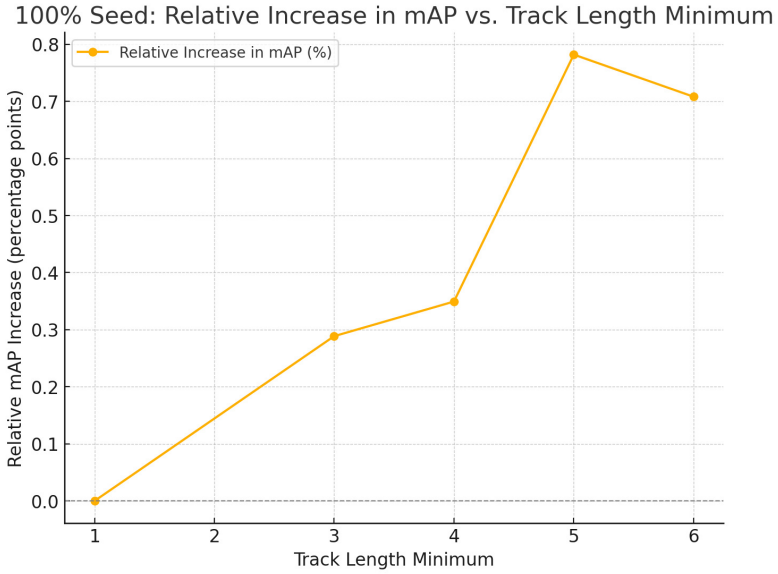


Figure 5.1.: Relative increase in mAP by filtering out ImmortalTracker's tracks with a low length for the 100% seed test case.

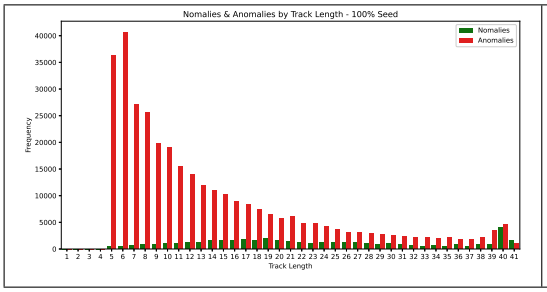
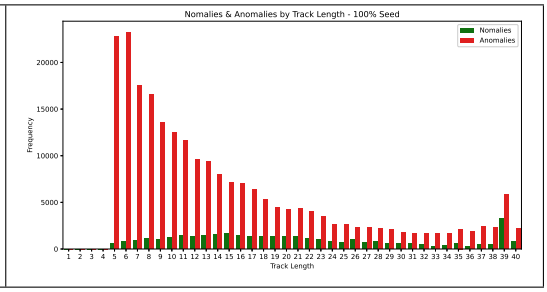
100% Seed Test Case	5% Seed Test Case
	

Table 5.2.: AnomalyAttention's training input showing anomalies and nomalies per track length, generated using 100% (left) and 5% (right) of the nuScene's training dataset.

For each test case, we train AnomalyAttention for a maximum of 200 epochs and a batch size of 24, which takes up to two days. During anomaly filtering, we apply confidence thresholds for each class, testing values of 0.5, 0.6, 0.7, 0.8, and 0.9 on the validation dataset, reflecting the sigmoid classification characteristic. For the Traffic-Cone class, where performance decreased across these thresholds, we set a threshold of 0.99, leading to minimal filtering but a slight performance improvement (Figures 5.7, 5.4).

The following sections outline the settings for the semi-supervised 5% seed test case and the fully supervised 100% seed test case. In both cases, anomalies relative to track length (Table 5.2) and relative to CenterPoint's detection score (Table 5.3) follow a tail distribution. This makes tracks of shorter length and detections with lower detection scores particularly relevant for anomaly filtering. We constrain AnomalyAttention's inference to specific track lengths and detection scores based on our model's performance on nuScenes' validation dataset. Typically, such evaluations should not be conducted on the test split; however, due to limited data and annotations, we aimed to approximate an optimal case as closely as possible for proof of concept.

5.1.1. 100% Seed Test Case

For this case, we evaluate our anomaly detection framework using the full training dataset. Our data-preprocessing framework (see Section 4.1) generates around 20k tracks with 37% anomalies (Positives) and 63% nomalies (Negatives) for the training dataset and around 37k tracks on the validation dataset with a minority of nomalies 11.3% (see Table 5.4). This class and dataset imbalance present major challenges for our model.

As longer tracks contain fewer anomalies (Figure 5.2) and result in reduced model performance (Figure 5.2), to optimize performance, we limit inference to tracks with lengths below 10. Similarly, inference is only performed on detections with detection scores below 0.2, as these represent the majority of anomalies (Figure 5.3) and exhibit a TNR performance close to 50% (Figure 5.3). All other detections are classified as

100% Seed Test Case	5% Seed Test Case

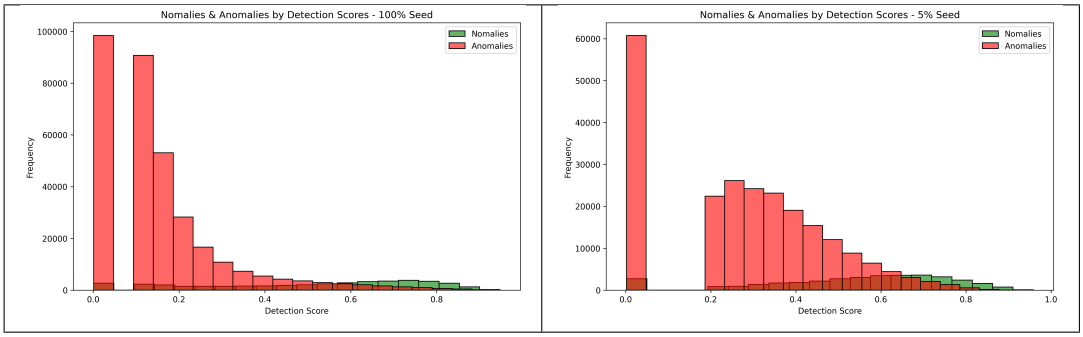


Table 5.3.: AnomalyAttention's training input showing anomalies and nomalies per detection score, generated using 100% (left) and 5% (right) of the nuScene's training dataset.

Dataset & Case	Tracks	Anomalies	Nomalies
Training & 100% Seed	19,748	112,307 (36.9%)	192,041 (63.1%)
Validation & 100% Seed	36,831	331,128 (88.7%)	42,279 (11.3%)
Training & 5% Seed & No Aug.	6,374	47,593 (86.3%)	7,528 (13.7%)
Training & 5% Seed	10,512	84,133 (81.6%)	18,950 (18.4%)
Validation & 5% Seed	24,969	230,264 (86.5%)	36,052 (13.5%)

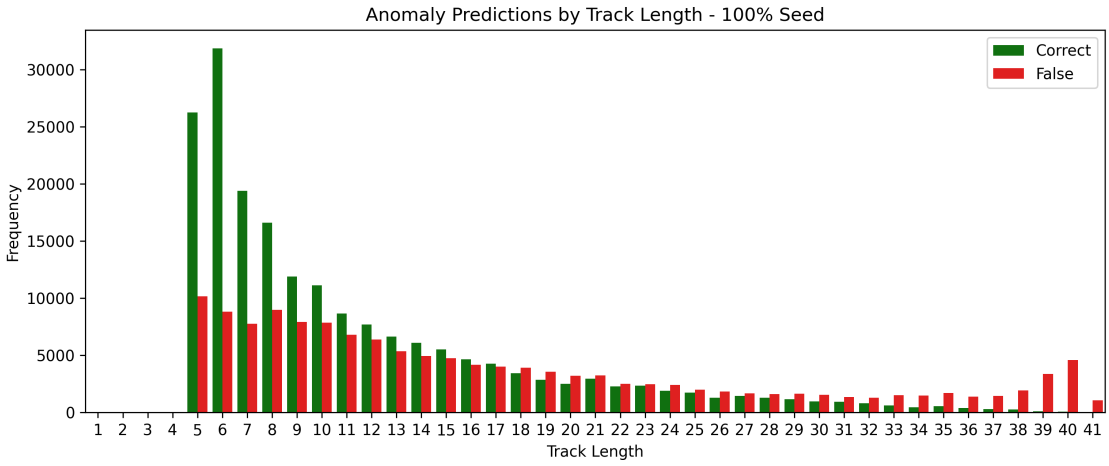
Table 5.4.: Summary of anomalies and nomalies generated for training and validation datasets with different seeding percentages. The "Training & 5% Seed No Aug." row highlights the distribution without data augmentation.

nomalies by our framework without further processing.

For optimal performance of AnomalyAttention, we use class-specific confidence thresholds, as detailed in Table 5.5. These thresholds vary by class, with a higher threshold (0.99) applied to traffic cones to minimize filtering and maintain accuracy (Figure 5.4), achieving the highest performance gains on the nuScenes validation dataset.

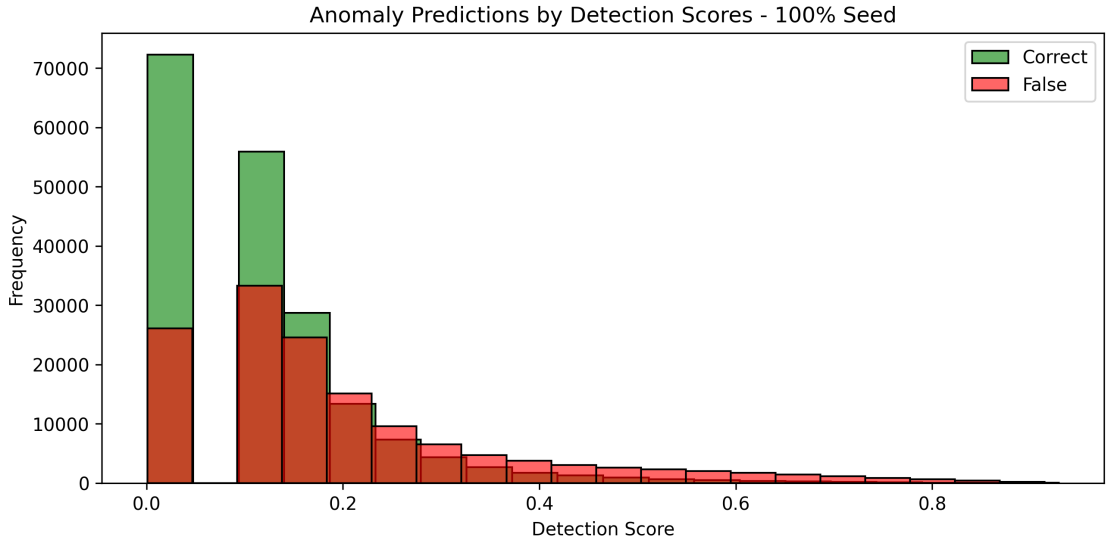
Model - Case	Bus	Trail.	Truck	Bicyc.	M-Cyc.	C-Veh.	Barrier	T-Cone
CenterPoint 5% seed - pseudo label generation	0.5	0.5	0.7	0.6	0.6	0.6	0.6	0.6
AnomalyAttention 5% seed - inference	0.6	0.7	0.6	0.6	0.6	0.7	0.9	0.99
AnomalyAttention 100% seed - inference	0.5	0.5	0.7	0.6	0.7	0.5	0.7	0.99

Table 5.5.: Confidence thresholds per class for each model - for CenterPoint's object detection and AnomalyAttention's anomaly detection. Abbreviations: Construction Vehicle (C-Veh.), Motorcycle (M-Cyc.), Bicycle (Bicyc.), Trailer (Trail.), and Traffic Cone (T-Cone).



Track Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
TPR	N/A	N/A	N/A	N/A	0.72	0.78	0.71	0.65	0.60	0.59	0.56	0.55	0.55	0.55	0.54	0.53	0.52	0.47	0.44	0.44	0.48	0.47	0.49	0.44	0.46	0.42	0.47	0.44	0.41	0.38	0.41	0.38	0.29	0.23	0.24	0.21	0.16	0.11	0.03	0.01	0.01
FPR	N/A	N/A	N/A	N/A	0.28	0.22	0.29	0.35	0.40	0.41	0.44	0.45	0.45	0.45	0.46	0.47	0.48	0.53	0.56	0.56	0.52	0.53	0.51	0.56	0.54	0.58	0.53	0.56	0.59	0.62	0.59	0.62	0.71	0.77	0.76	0.79	0.84	0.89	0.97	0.99	0.99

Figure 5.2.: AnomalyAttention's performance on nuScene's valiation dataset, trained 100% seed data, showing predicted anomalies per track length.



ScoreBin	0.0–0.1	0.1–0.1	0.1–0.2	0.2–0.2	0.2–0.2	0.2–0.3	0.3–0.4	0.4–0.4	0.4–0.5	0.5–0.5	0.5–0.6	0.6–0.6	0.6–0.7	0.7–0.7	0.7–0.8	0.8–0.8	0.8–0.9	0.9–0.9	0.9–1.0	1.0–1.0
TPR	0.74	N/A	0.61	0.52	0.46	0.43	0.38	0.35	0.30	0.28	0.23	0.19	0.19	0.18	0.16	0.16	0.11	0.15	0.24	N/A
FPR	0.26	N/A	0.39	0.48	0.54	0.57	0.62	0.65	0.70	0.72	0.77	0.81	0.81	0.82	0.84	0.84	0.89	0.85	0.76	N/A

Figure 5.3.: AnomalyAttention's performance on nuScene's valiation dataset, trained 100% seed data, showing predicted anomalies per detection score.

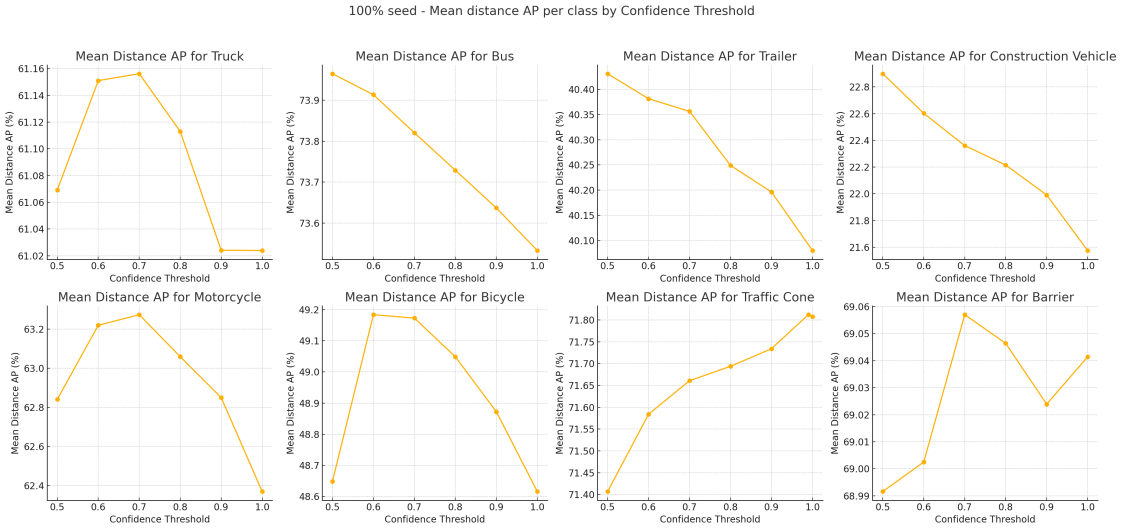


Figure 5.4.: AnomalyAttention's performance, trained on 100% seed data, by confidence score per class on the nuScenes validation dataset.

5.1.2. 5% Seed Test Case - Semi-Supervised Learning

In contrast to the 100% seed case (Section 5.1.1), we evaluate our framework using only 5% of the training dataset as labeled data. This subset consists of 35 scenes, sampled to maintain a similar distribution to the full dataset Table 5.1. CenterPoint, trained on this 5% seed dataset (Table 5.4), is used to predict pseudo labels for the remaining 95% of the training data, which is treated as unlabeled.

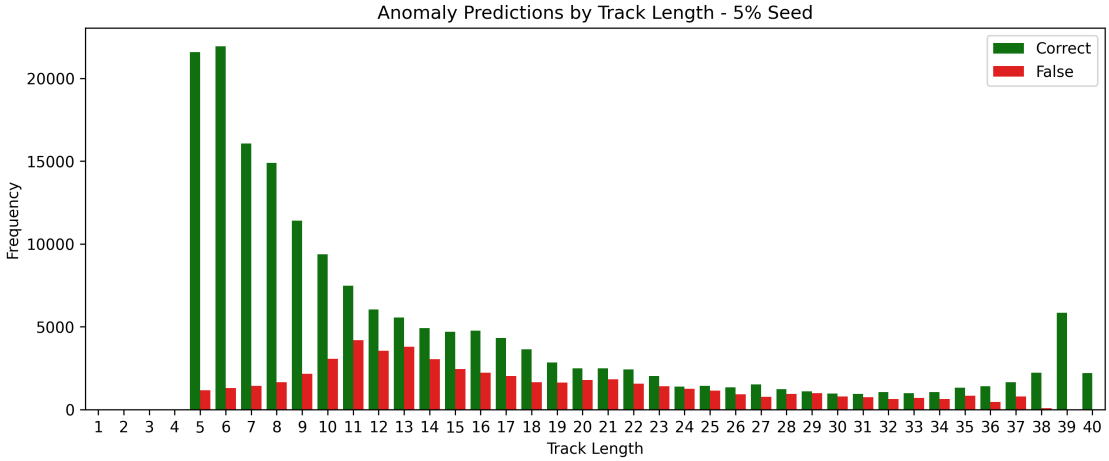
For this pseudo-labeling process, we apply class-dependent confidence thresholds (Table 5.5 for CenterPoint's detection scores to balance the recall-precision trade-off for each class (Appendix B). After pseudo-labeling, CenterPoint is retrained from scratch using both the 5% labeled seed data and the 95% pseudo-labeled data for 20 epochs. Detections are then further processed as descibed in Section 4.1.

To address the limited data availability, we supplemented the dataset by generating additional anomalous tracks using nuScenes ground truth detection tracks corresponding to the 5% seed dataset. Since these tracks are independent of CenterPoint's detection performance, we apply extensive augmentations (see Section 4.1.2). Each ground truth track is sampled twice, with its detections having an 80% chance of being augmented into an anomaly, designed to match the anomaly distribution of real samples (see Section 4.1.2). This resulted in a training dataset with approximately 11k tracks, containing 81% anomalies and 18% nomalies—closely matching the validation dataset distribution of 25k tracks (Table 5.4). While the datasets have no significant imbalance in their distributions compared to the 100% seed case, the inherent class imbalance remains a significant challenge for our model.

For our data preprocessing pipeline (Figure 4.1) on the nuScenes validation dataset,

we observed that anomalies dominate across all track lengths (Figure 5.2). To improve performance, we restricted AnomalyAttention's inference by applying stricter thresholds for track length and detection scores, as it was trained on fewer real samples in this case. Due to lower model performance (approximately 70% TNR; Figure 5.5), we excluded AnomalyAttention's inference for tracks with lengths between 11 and 35 and for detections with scores above 0.5.

Similarly to the 100% Seed Test case (Section 5.1.1), anomalies are more prevalent at lower detection scores (Figure 5.3). Since AnomalyAttention's performance declines at higher detection scores, we limited its inference to detection in tracks with scores below 0.6 (approximately 66% TNR; Figure 5.6).



Track Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
TPR	N/A	N/A	N/A	N/A	0.95	0.94	0.92	0.90	0.84	0.75	0.64	0.63	0.60	0.62	0.66	0.68	0.68	0.69	0.64	0.58	0.58	0.61	0.59	0.52	0.56	0.59	0.66	0.57	0.53	0.55	0.56	0.62	0.59	0.62	0.61	0.76	0.68	0.96	1.00	1.00
FPR	N/A	N/A	N/A	N/A	0.05	0.06	0.08	0.10	0.16	0.25	0.36	0.37	0.40	0.38	0.34	0.32	0.32	0.31	0.36	0.42	0.42	0.39	0.41	0.48	0.44	0.41	0.34	0.43	0.47	0.45	0.44	0.38	0.41	0.38	0.39	0.24	0.32	0.04	0.00	0.00

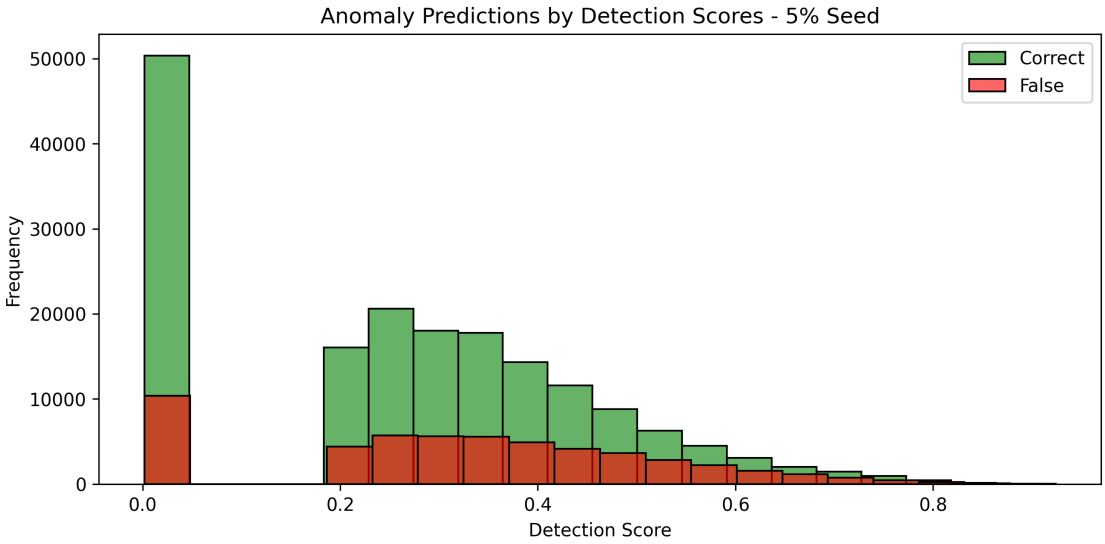
Figure 5.5.: AnomalyAttention's performance on nuScene's valiation dataset, trained 5% seed data, showing predicted anomalies per track length.

As for the 100% seed case, we tested various confidence thresholds for each class to achieve optimal performance. The final confidence thresholds for the 5% seed validation case are shown in Table 5.5 and were selected based on the results in Figure 5.7.

5.2. Training Results

In this section, we summarize the results on the development validation dataset, which represents 15% of the respective seed dataset used to monitor training. This dataset is distinct from the validation dataset used for final evaluation, as discussed in other sections.

We began our development with the 5% seed dataset case, evaluating several model approaches. For completeness, results from these intermediate models are included in Table 5.6. One approach, based on PointNet (Section 4.3.2), failed to achieve accuracy



ScoreBin	0.0–0.1	0.1–0.1	0.1–0.2	0.2–0.2	0.2–0.2	0.2–0.3	0.3–0.4	0.4–0.4	0.4–0.5	0.5–0.5	0.5–0.6	0.6–0.6	0.6–0.7	0.7–0.7	0.7–0.8	0.8–0.8	0.8–0.9	0.9–0.9	0.9–1.0	1.0–1.0
TPR	0.83	N/A	N/A	N/A	0.80	0.77	0.77	0.75	0.74	0.71	0.69	0.66	0.64	0.62	0.65	0.63	0.56	0.59	0.30	N/A
FPR	0.17	N/A	N/A	N/A	0.20	0.23	0.23	0.25	0.26	0.29	0.31	0.34	0.36	0.38	0.35	0.37	0.44	0.41	0.70	N/A

Figure 5.6.: AnomalyAttention's performance on nuScene's valiation dataset, trained 5% seed data, showing predicted anomalies per detection score.

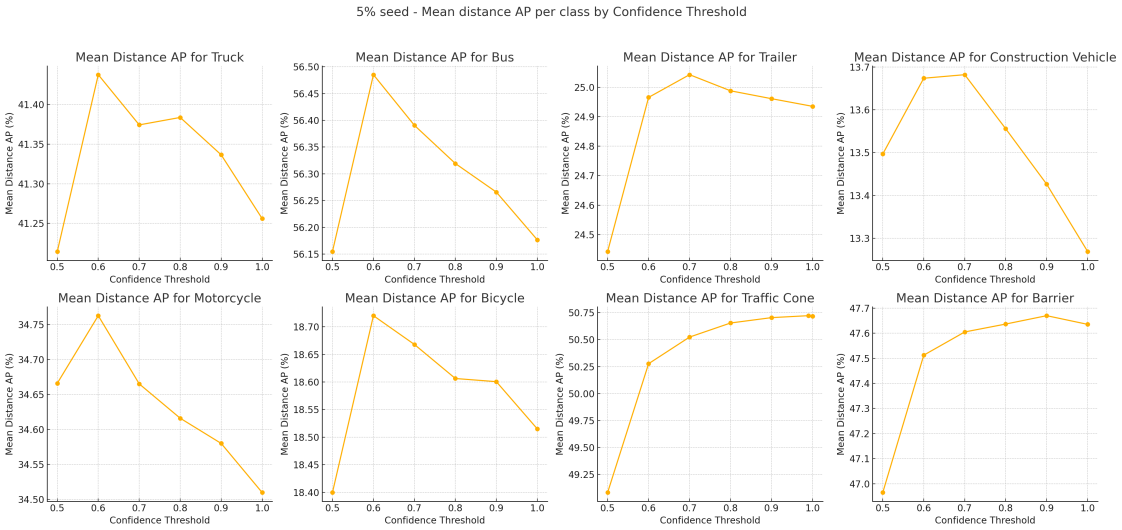


Figure 5.7.: AnomalyAttention's performance, trained on 5% seed data, by confidence score per class on the nuScenes validation dataset.

above 50% and was excluded from further experiments prior to making modifications to the input structure and loss function.

AnomalyAttention demonstrated the highest potential in overall accuracy and anomaly detection (TPR), attributed to its high capacity. However, the representation of these results is constrained by the limited amount of available data.

Table 5.6.: Results for different anomaly detectors on the development validation dataset (15% of the training dataset). Abbreviation: Augmentation (Aug.) and Balanced Accuracy (B-Acc)

Model & Seed Size	Accuracy	B-Acc	TPR	TNR	FPR	FNR	Anomalies	Nomalies	Parameters	Size in MB
ResNetAD 5%	61.01	60.37	66.10	54.63	45.37	33.90	1,507	7,850	51k	0.2
VoxelNetAD 5%	61.90	52.70	70.23	35.17	64.83	29.77	1,507	7,850	13 M	49.26
Training & 5% Seed & No Aug.	66.23	60.92	67.23	54.60	45.40	32.77	1,507	7,850	19 M	72.37
AnomalyAttention 5%	71.74	58.35	76.71	39.98	60.02	23.29	1,507	7,850	19 M	72.37
AnomalyAttention 100%	70.01	65.36	44.18	86.53	13.47	55.82	24,685	15,784	19 M	72.37

5.3. Quantitative Results

In this section, give high-level quantative results for our anomaly detector and the overall framework. In summary:

- **Binary Classification Results:** AnomalyAttention achieves high Precision (98.20% for 100% seed, 97.94% for 5% seed) and TNR (94.94% and 97.28%, respectively), effectively retaining nomalies. However, the model struggles with detecting anomalies, with Recall at 35.18% for 100% seed and 20.29% for 5% seed.
- **100% Seed Case:** Our framework slightly improves mAP by +0.87%, with notable gains in rare classes like Motorcycle (+1.25%), Construction Vehicle (+1.69%), and Bicycle (+2.75%).
- **5% Seed Case:** The semi-supervised approach yields a significant mAP improvement of +6.67%, with strong performance in Bus (+9.52%) and other rare classes.

5.3.1. AnomalyAttention

We evaluate AnomalyAttention's performance for each test case using binary classification metrics (Section 2.1). The results (Table 5.7) demonstrate a trade-off between anomaly detection and nomaly retention, with the model prioritizing high Precision and True Negative Rate (TNR). For both test cases, most anomalies are not detected, while the majority of nomalies are correctly identified (Figures 5.8 and 5.9). This behavior reflects the strict thresholds employed, which are designed to minimize false positives, ensuring that nomalies are retained in high numbers.

For the 100% seed test case, AnomalyAttention detects 35% of anomalies (TPR) while misclassifying 5% of nomalies (FPR). In contrast, for the 5% seed test case, the model

detects only 20% of anomalies (TPR) but misclassifies just 3% of nomalies (FPR). These results illustrate the model's capacity to retain almost all nomalies, even under semi-supervised conditions, while achieving reasonable anomaly detection rates given the limited training data in the 5% seed case.

The lower recall for anomalies in the 5% seed case (20.29%) highlights the challenges posed by reduced labeled data and suggests room for improvement in our data augmentation strategy and threshold settings. However, the high Precision (97.94%) and TNR (97.28%) confirm the model's ability to maintain reliability in detecting nomalies, even in resource-constrained scenarios.

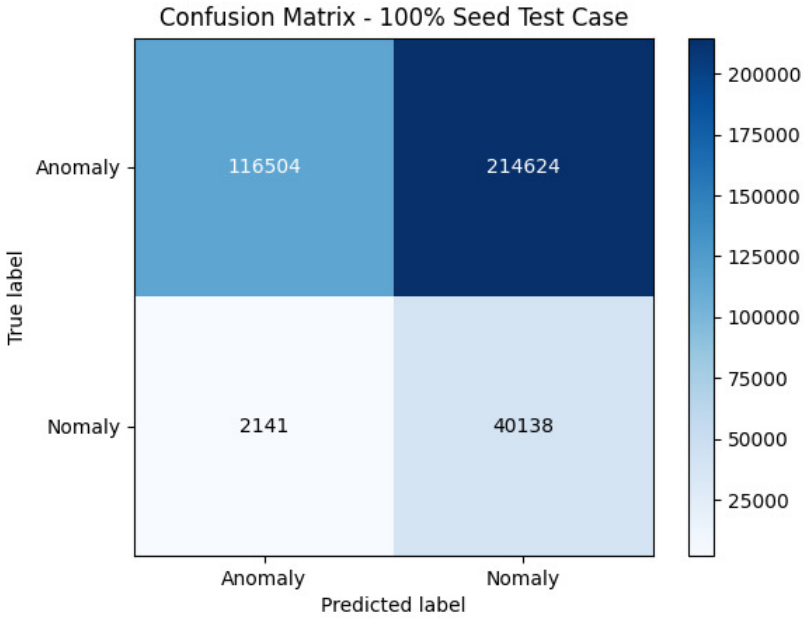


Figure 5.8.: Confusion matrix showing AnomalyAttention's performance on the nuScenes validation dataset for the 100% seed test case.

Table 5.7.: Binary classification metrics for AnomalyAttention's performance on the nuScenes validation dataset for the 5% and 100% seed test cases. *Positive* (P) is an anomaly and a *Negative* (N) is a nomaly. Balanced Accuracy (B-Acc)

Test Case	B-Acc	Precision	Recall	TNR	FPR	FNR	TP	TN	FP	FN
100% Test Case	65.06	98.20%	35.18%	94.94%	5.06%	64.82%	116504	40138	2141	214624
5% Test Case	58.79	97.94%	20.29%	97.28%	2.72%	79.71%	46725	35070	982	183539

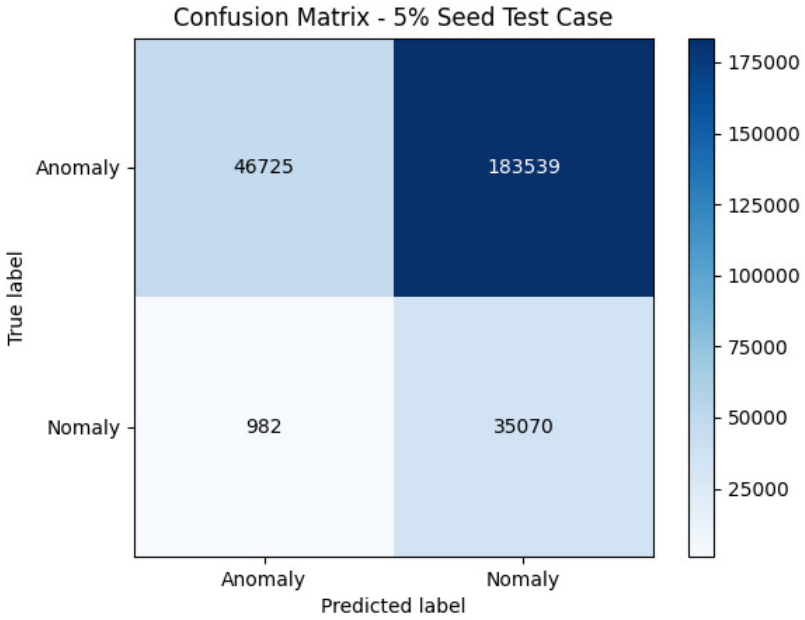


Figure 5.9.: Confusion matrix showing AnomalyAttention's performance on the nuScenes validation dataset for the 5% seed test case.

5.3.2. Overall Framework

Evaluating the whole framework using nuScenes' object detection benchmark reveals distinct patterns (Table 5.8). For the 100% seed test case, we observe a limited improvement of +0.87% mAP, with significant gains in specific low-performing classes, including Motorcycle (+1.25%), Construction Vehicle (+1.69%), and Bicycle (+2.75%). These improvements are particularly noteworthy as they target rare or challenging object categories where the baseline (CenterPoint) struggles.

For the 5% seed test case, the semi-supervised approach yields a strong +6.67% mAP improvement, with gains across all classes. The Bus class has a strong improvement of +9.52%, underscoring the effectiveness of the framework in addressing data-scarce scenarios. These results highlight the potential of semi-supervised learning to bridge the performance gap between fully-supervised scenarios and settings with limited labeled data. More details of qualitative results for different pipeline steps of the framework are discussed for both cases in our ablation studies (Section 5.5).

5.4. Qualitative Results

In this section, we present qualitative results for the nuScenes validation dataset in bird's-eye view to illustrate the impact of AnomalyAttention and our overall framework. More examples are provided in the Appendix D.

Table 5.8.: Class-wise Average Precision (AP) comparison between our framework and CenterPoint on the nuScenes validation dataset. Abbreviations: Construction Vehicle (C-Veh.), Motorcycle (M-Cyc.), Bicycle (Bicyc.), Trailer (Trail.), and Traffic Cone (T-Cone).

Model	**mAP**	**Truck**	**Bus**	**Trail.**	**C-Veh.**	**M-Cyc.**	**Bicyc.**	**T-Cone**	**Barrier**
CenterPoint 100%	55.36	61.00	74.00	39.34	20.12	62.02	45.90	71.47	69.04
+ Ours	56.23	61.06	73.97	40.17	21.81	63.27	48.65	71.81	69.06
+/-	**+0.87**	**+0.06**	**-0.03**	**+0.83**	**+1.69**	**+1.25**	**+2.75**	**+0.41**	**+0.02**
CenterPoint 5%	29.40	35.29	46.96	18.38	10.68	28.09	10.41	44.87	40.50
+ Ours	36.07	41.44	56.48	25.04	13.68	34.76	18.72	50.72	47.67
+/-	**+6.67**	**+6.09**	**+9.52**	**+6.66**	**+3.00**	**+6.67**	**+8.31**	**+5.85**	**+7.17**

5.4.1. AnomalyAttention

From the examples, AnomalyAttention demonstrates its ability to effectively filter anomalies, particularly in the 5% seed test case. Notably, it is more effective in removing anomalies for larger objects, such as trucks (Table 5.9). Interestingly, some anomalies, such as a misdetected truck in the top-right of the example frame in Table 5.10, are consistently filtered in both test cases. This consistency suggests that AnomalyAttention successfully learns an anomaly paradigm, enabling it to generalize across different training scenarios.

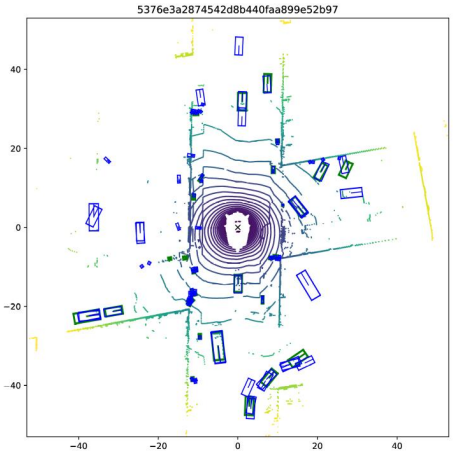
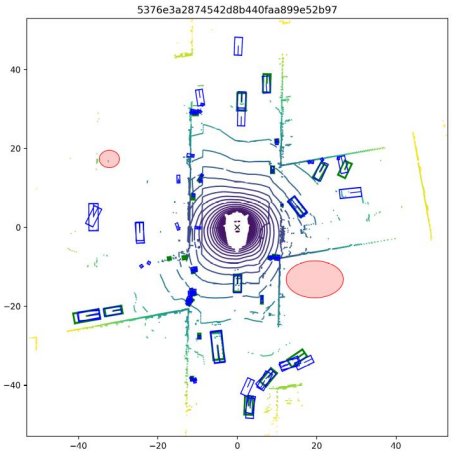
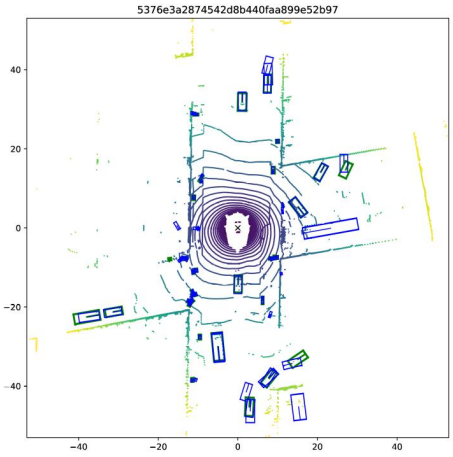
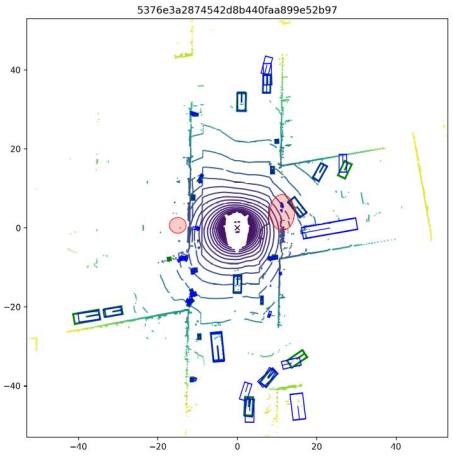
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table 5.9.: Qualitative Results for nuScenes frame 5376e3a2874542d8b440faa899e52b97 without AnomalyAttention compared to with AnomalyAttention. Predicted bounding boxes are (blue), ground truth bounding boxes are (green), and filtered areas are (red).

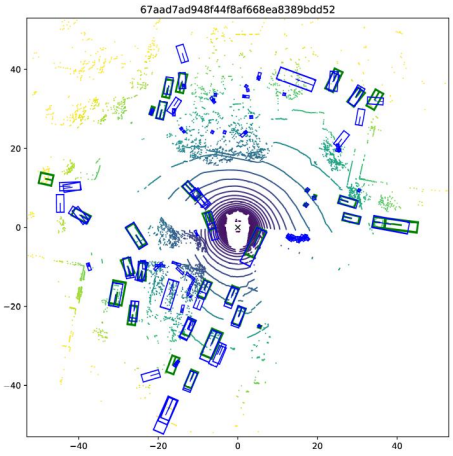
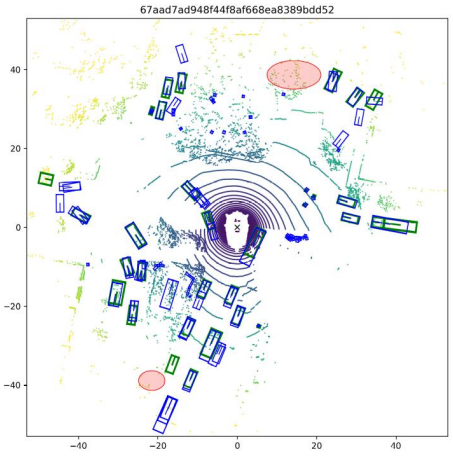
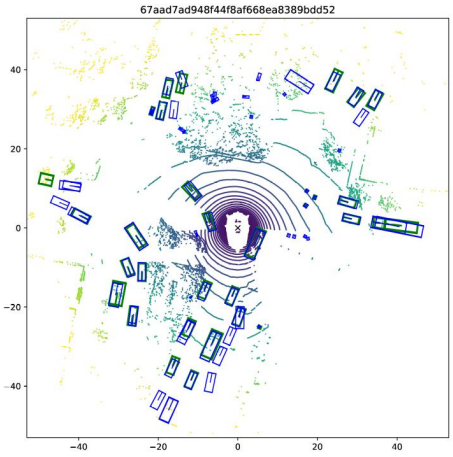
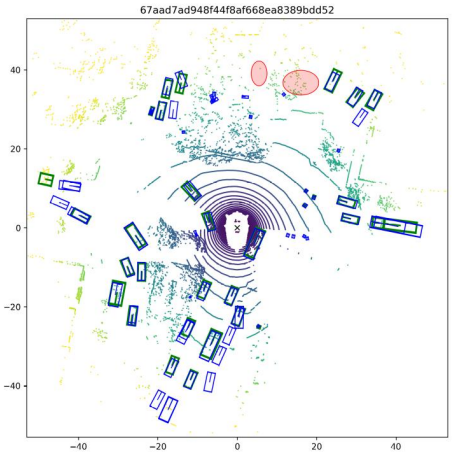
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table 5.10.: Qualitative Results for nuScenes frame 67aad7ad948f44f8af668ea8389bdd52 without AnomalyAttention compared to with AnomalyAttention. Predicted bounding boxes are (blue), ground truth bounding boxes are (green), and filtered areas are (red).

5.4.2. Overall Framework

Comparing outputs between the baseline model (CenterPoint) and our framework, we observe a significant reduction in false detections in both test cases. Specifically, incorrect bounding boxes of larger sizes and overlapping boxes are effectively removed (Table 5.11), underscoring the qualitative strength of our framework. These improvements are especially evident in the 5% seed test case, where limited labeled data amplifies the benefits of anomaly filtering , and targeted preprocessing. However, we also observe new false-positives introduced by from training with pseudo-labels (Table 5.12).

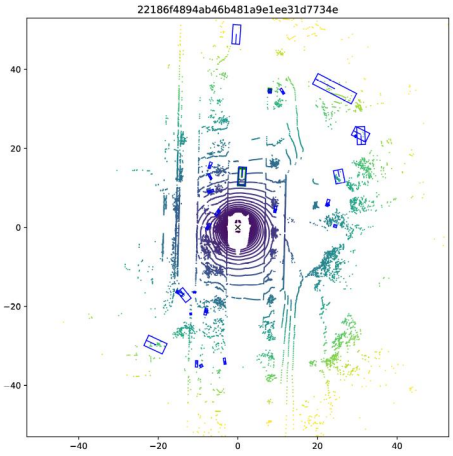
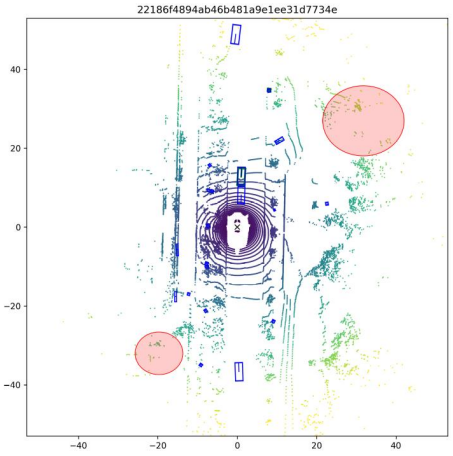
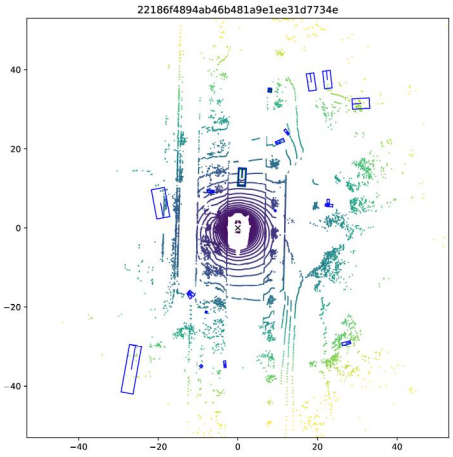
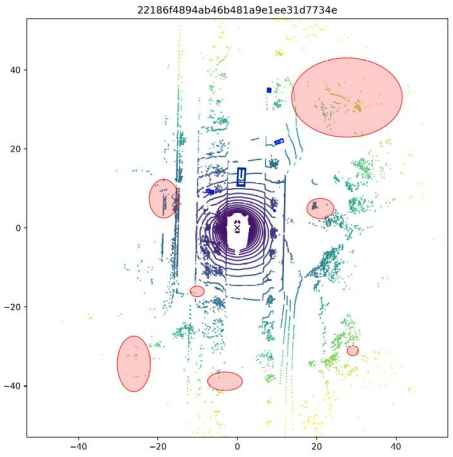
CenterPoint (5% Seed)	CenterPoint+Ours (5% Seed)
	
CenterPoint (100% Seed)	**CenterPoint+Ours (100% Seed)**
	

Table 5.11.: Qualitative Results for nuScenes frame 22186f4894ab46b481a9e1ee31d7734e for CenterPoint and CenterPoint+Ours. Predicted bounding boxes are (blue), ground truth bounding boxes are (green), and areas with significant change are (red).

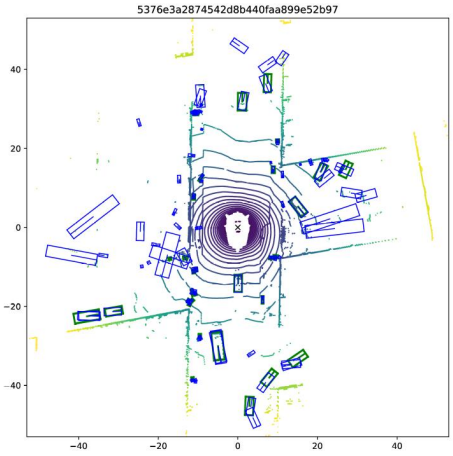
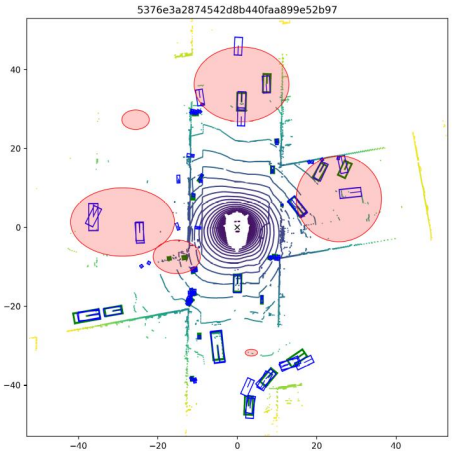
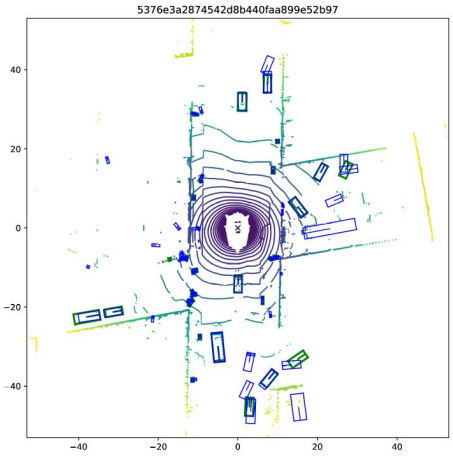
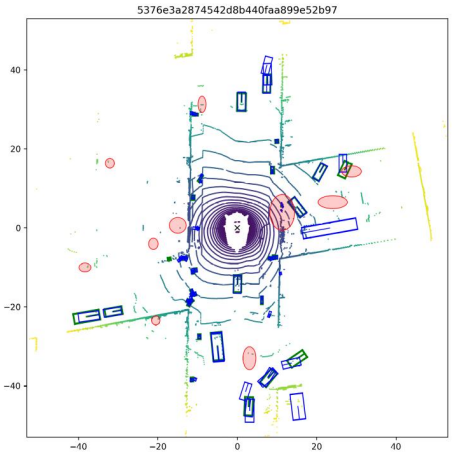
CenterPoint (5% Seed)	CenterPoint+Ours (5% Seed)
	
CenterPoint (100% Seed)	**CenterPoint+Ours (100% Seed)**
	

Table 5.12.: Qualitative Results for nuScenes frame 5376e3a2874542d8b440faa899e52b97 for CenterPoint and CenterPoint+Ours. Predicted bounding boxes are (blue), ground truth bounding boxes are (green), and areas with significant change are (red).

5.5. Ablation Studies

In this section, we analyze the contribution of each module in our framework to the overall performance. By evaluating the individual and cumulative impact of key pipeline components, we provide insights into their effectiveness for both the 100% seed and 5% seed test cases.

5.5.1. 100% Seed Test Case

For the 100% seed test case, each module contributes incrementally to the overall performance (Table 5.13):

- **ImmortalTracker** introduces additional detections by extending object tracks, but this also increases false positives significantly (+800k), leading to a slight drop in mAP (-0.14%).
- **Filter tracks < 5:** Filtering tracks shorter than 5 frames mitigates this noise, reducing false positives (-1M) and improving mAP by +0.78%.
- **AnomalyAttention** further enhances the results by refining false positives (-160k), contributing an additional mAP boost of +0.23%. However, its overall impact is constrained by its performance limitations, the disproportionate importance of True Positives for mAP, and the resulting need for strict thresholds to preserve nomalies. These thresholds, while effective at retaining CenterPoint's TP detections, inherently limit the detection of false positives (FP), which hinders AnomalyAttention's ability to significantly boost performance in this fully-supervised scenario.

Table 5.13.: Detailed performance metrics for both test cases on the nuScenes validation dataset.

Pipeline	mAP	TP	FP	FN
CenterPoint	55.36	158,181	1,039,227	42,807
+ ImmortalTracker	55.22 (-0.14%)	162,522 **(+4k)**	1,899,114 (+800k)	38,466 (-4k)
+ Filter tracks < 5	56.00 **(+0.78%)**	158,052 (-4k)	867,760 **(-1Mil)**	42,936 (+4k)
+ AnomalyAttention	56.23 (+0.23%)	157,638 (-400)	698,350 (-160k)	43,350 (+400)
Total	**+0.87%**	**-543**	**-340,877**	**+543**
CenterPoint – 5% seed	29.40	136,034	1,815,844	84,972
+ 95 pseudo training	34.84 **(+5.44%)**	142,411 (+6k)	694,279 **(-1,1M)**	78,595 (-6k)
+ ImmortalTracker	34.98 (+0.14%)	152,245 **(+10k)**	1,316,273 (+600k)	68,761 (-10k)
+ Filter tracks < 5	35.88 (+0.90%)	144,487 (-8k)	610,787 (-700k)	76,519 (+8k)
+ AnomalyAttention	36.07 (+0.19%)	143,889 (-600)	485,205 (-126k)	77,117 (+600)
Total	**+6.67%**	**+7,855**	**-1,330,639**	**-7,855**

Class-wise mAP analysis (Table 5.14) highlights AnomalyAttention's limitations, with its contribution being comparably lower than the gains achieved through track length

filtering. This underscores the value of preprocessing steps in reducing noise and enhancing overall framework performance.

Table 5.14.: Class-wise AP comparisons of different pipeline stages for both test cases on the nuScenes validation dataset. Abbreviations: Construction Vehicle (C-Veh.), Motorcycle (M-Cyc.), and Traffic Cone (T-Cone).

Pipeline	mAP	Truck	Bus	Trailer	C-Veh.	M-Cyc.	Bicycle	T-Cone	Barrier
CenterPoint 100%	55.36	61.00	74.00	39.34	20.12	62.02	45.90	71.47	69.04
+ ImmortalTracker	55.22 (-0.14%)	60.78 (-0.22)	73.21 (-0.79)	39.33 (-0.01)	20.25 (+0.13)	61.76 (-0.2)	45.92 (+0.02)	71.52 (+0.05)	69.01 (-0.03)
+ Filter tracks < 5	56.01 (+**0.79%**)	61.02 (+**0.24**)	73.53 (+0.32)	40.08 (+**0.75**)	21.57 (+**1.32**)	62.37 (+0.61)	48.62 (+**2.70**)	71.81 (+**0.29**)	69.04 (+**0.03**)
+ AnomalyAttention	56.23 (+0.23%)	61.06 (+0.04)	73.97 (+**0.44**)	40.17 (+0.09)	21.81 (+0.24)	63.27 (+**0.90**)	48.65 (+0.03)	71.81 (+0.00)	69.06 (+0.02)
Total	**+0.87%**	**+0.06**	**-0.03**	**+0.83**	**+1.69**	**+1.25**	**+2.75**	**+0.34**	**+0.02**
CenterPoint 5%	29.40	35.29	46.96	18.38	10.68	28.09	10.41	44.87	40.50
+ Pseudo Training	34.84 (+**5.44%**)	40.97 (+**5.68**)	56.23 (+**9.27**)	23.35 (+**4.97**)	10.95 (+0.27)	34.88 (+**6.79**)	15.66 (+**5.25**)	49.76 (+**4.89**)	46.91 (+**6.41**)
+ ImmortalTracker	34.98 (+0.14%)	40.88 (-0.09)	55.76 (-0.47)	23.80 (+0.45)	11.10 (+0.15)	34.88 (+0.00)	15.67 (+0.01)	50.66 (+0.90)	47.12 (+0.21)
+ Filter tracks < 5	35.88 (+0.90%)	41.26 (+0.38)	56.18 (+0.42)	24.94 (+1.14)	13.27 (+**2.17**)	34.51 (-0.37)	18.52 (+**2.85**)	50.72 (+0.06)	47.64 (+0.52)
+ AnomalyAttention	36.07 (+0.19%)	41.44 (+0.18)	56.49 (+0.31)	25.04 (+0.10)	13.68 (+0.41)	34.76 (+0.25)	18.72 (+0.20)	50.72 (+0.00)	47.67 (+0.03)
Total	**+6.67%**	**+6.15**	**+9.53**	**+6.66**	**+3.00**	**+6.67**	**+8.31**	**+5.85**	**+7.17**

5.5.2. 5% Seed Test Case

For the 5% seed test case, each pipeline module demonstrates a substantial impact on performance, underscoring the value of semi-supervised learning and targeted preprocessing (Table 5.13):

- **Pseudo Training:** Beginning with a baseline mAP of 29.40%, the addition of pseudo training from 95% of the dataset results in a remarkable mAP boost of +5.44%, largely due to a significant reduction in false positives (-1.1M) and an increase in true positives (+6k). This highlights the efficacy of leveraging pseudo-labeled data to mitigate the limitations of small seed datasets.

- **ImmortalTracker** improves true positives (+10k) but increases false positives (+600k), yielding only a marginal mAP improvement of +0.14%. This demonstrates ImmortalTracker's ability to extend object tracks but also its susceptibility to introducing noise, particularly for challenging classes such as Trailer (+0.45%) and Traffic Cone (+0.90%).

- **Filter tracks < 5:** Filtering tracks shorter than 5 frames mitigates much of the noise, reducing false positives by 700k and delivering a notable mAP gain of +0.90%. Class-wise, this step provides substantial improvements for rare categories such as Bicycle (+2.85%), Construction Vehicle (+2.17%), and Trailer (+1.14%).

- **AnomalyAttention** refines the results further by reducing false positives (-126k) and contributing an additional mAP increase of +0.19%. While its contribution is modest, it remains valuable for boosting challenging classes like Motorcycle (+0.25%), Bicycle (+0.20%), and Construction Vehicle (+0.41%). However, its impact is limited by the reliance on strict thresholds to preserve true positives, a constraint necessary for maintaining high mAP values.

Class-wise performance (Table 5.14) mirrors these trends. The pseudo training step provides the most significant gains across all classes, especially Bus (+9.27%), Trailer (+4.97%), and Motorcycle (+6.79%). The cumulative effect of all modules results in a total mAP improvement of +6.67%, with the largest gains observed for underrepresented and challenging categories, highlighting the robustness of the framework in addressing data scarcity and class imbalance.

5.5.3. Impact of Distance Threshold (4m)

To align with the anomaly definitions used in our training and evaluation for Anomaly-Attention, we also considered mAP performance respecting only the 4-meter distance threshold (Table 5.15). While an overall performance increase is observed, as expected due to using only the loosest threshold, no significant differences are found in the behavior and impact of AnomalyAttention.

Table 5.15.: Performance comparison of different pipeline stages for 4 meters distance for both test cases on the nuScenes validation dataset. Abbreviations: Construction Vehicle (C-Veh.), Motorcycle (M-Cyc.), and Traffic Cone (T-Cone).

Pipeline	mAP - 4m	Truck	Bus	Trailer	C-Veh.	M-Cyc.	Bicycle	T-Cone	Barrier
CenterPoint 100%	64.73	71.39	86.68	61.13	38.57	64.93	46.52	74.96	73.69
Immortal	64.89 (+0.16)	71.73 (+0.34)	87.37 (+0.69)	61.08 (-0.05)	38.62 (+0.05)	64.93 (+0.00)	46.52 (+0.00)	74.99 (+0.03)	73.87 (+0.18)
Filter tracks < 5	65.97 (+1.08)	72.10 (+0.37)	87.95 (+0.58)	62.25 (+1.17)	41.23 (+2.61)	65.63 (+0.70)	49.34 (+2.82)	75.27 (+0.28)	73.98 (+0.11)
AnomalyAttention	66.14 (+0.17)	72.06 (-0.04)	88.29 (+0.34)	62.38 (+0.13)	41.79 (+0.56)	65.93 (+0.30)	49.38 (+0.04)	75.27 (+0.00)	74.01 (+0.03)
Total	**+1.41**	**+0.67**	**+1.61**	**+1.25**	**+3.22**	**+1.00**	**+2.86**	**+0.31**	**+0.32**
CenterPoint 5%	38.05	46.34	65.60	28.66	24.70	31.83	10.80	49.91	49.55
+ Pseudo Training	43.89 (+5.84)	49.85 (+3.51)	73.30 (+7.70)	35.53 (+6.87)	23.09 (-1.61)	38.58 (+6.75)	16.12 (+5.32)	54.95 (+5.04)	53.74 (+4.19)
+ ImmortalTracker	44.32 (+0.43)	50.23 (+0.38)	73.79 (+0.49)	36.42 (+0.89)	23.57 (+0.48)	38.59 (+0.01)	16.12 (+0.00)	55.91 (+0.96)	53.96 (+0.22)
+ Filter tracks < 5	44.83 (+0.51)	50.67 (+0.44)	74.38 (+0.59)	38.37 (+1.95)	27.81 (+4.24)	38.04 (-0.55)	19.06 (+2.94)	55.75 (-0.16)	54.59 (+0.63)
+ AnomalyAttention	45.09 (+0.2)	50.91 (+0.24)	74.88 (+0.50)	38.37 (+0.00)	28.55 (+0.74)	38.32 (+0.28)	19.29 (+0.23)	55.76 (+0.01)	54.64 (+0.05)
Total	**+7.04**	**+4.57**	**+9.28**	**+9.71**	**+3.85**	**+6.49**	**+8.49**	**+5.85**	**+5.09**

6. Conclusion & Future Work

In this section, we summarize the main conclusions of our work and propose ideas for potentially impactful future research directions.

6.1. Conclusion

This thesis demonstrates the effectiveness of our proposed framework for anomaly detection in LiDAR-based object detection systems, evaluated on the challenging nuScenes dataset [6]. By integrating multiple modular components, including CenterPoint [60], ImmortalTracker [55], track-length filtering, and our AnomalyAttention module, we achieve improvements in precision and false-detection filtering across both fully supervised and semi-supervised test cases.

Our results highlight the following key findings:

- **Modular Contributions**: Each module in our pipeline uniquely enhances performance. Pseudo training (for the semi-supervised case) addresses data scarcity, ImmortalTracker extends object detections across frames, track-length filtering reduces noise, and AnomalyAttention refines anomalies with a focus on rare or challenging classes.

- **Semi-Supervised vs. Fully Supervised**: Using only 5% of the training dataset, our semi-supervised case demonstrates the robustness and scalability of our framework in handling limited labeled data. Pseudo training provides the most significant mAP improvement, while track-length filtering and AnomalyAttention complement it by reducing false object detections. However, for both test cases, the overall precision improvement from anomaly filtering lies below 1%, indicating that its limitations are not primarily due to data scarcity.

- **Class-Specific Insights**: There is research [8] that reports that pseudo-labeling tends to favor majority classes, reducing detection accuracy for minority classes. However, our experiments reveal the opposite effect in 3D detection, where tail object classes, such as Construction Vehicle, Bicycle, and Motorcycle, benefit the most from the proposed pipeline. These gains highlight the framework's effectiveness in improving detection for underrepresented categories.

- **Robust Feature Representation**: Our anomaly detector, *AnomalyAttention*, leverages LSTM-based track features, VoxelNet for point-cloud features, and attention-based feature fusion. This architecture effectively combines temporal and spatial

information to identify anomalies while retaining valid detections. Although the overall mAP boost is limited, the trade-off of losing 400 true object detections to remove 160k false positives (Table 5.13) is a worthwhile compromise for accelerating 3D data labeling. Additionally, this approach generates examples that enable further analysis of wrong behavior by the base detector.

6.2. Future Work

Our findings can pave the way for future work in anomaly detection and refinement for LiDAR-based systems, particularly in addressing class imbalance and enhancing spatial precision. Our framework can serve as a foundation for further advancements in autonomous driving and other applications relying on precise object detection in complex environments. We see three main domains that could be addressed in future work:

1. **Model:** Future studies could focus on detailed hyperparameter tuning and architecture optimization to identify the most valuable components of our model and enhance overall performance. For instance, increasing dropout rates might improve generalization and pruning less effective elements of the architecture could improve both efficiency and accuracy. Testing further alternative inference scopes systematically, particularly for track lengths and detection scores, could uncover additional performance gains.

2. **Data Modality and Preprocessing:** Enhancements in data handling and preprocessing could unlock further improvements in the framework. Context-aware padding values, tailored to individual features, might yield better feature representation compared to the uniform padding used in this study. Additionally, some components of the architecture, such as LSTMs, naturally support varying input lengths, making it possible to bypass padding entirely in those areas, thus reducing computational overhead. Sampling LiDAR points specifically around detections rather than from the entire point cloud would allow higher resolution for bounding boxes and improve spatial precision. Exploring alternative augmentation techniques, integrating other data modalities (e.g., images), and revising strict track-length filtering criteria could yield better results. Applying the framework to datasets like Waymo Open Dataset, which offers higher annotation density and longer tracks, could also provide additional insights and improvements.

3. **Evaluation and Metrics:** Defining metrics that better quantify the practical benefits of anomaly filtering remains an important area for future research. For instance, metrics designed to reflect reduced human workload for 3D annotations could better capture the trade-offs between retaining true positives and eliminating false positives, which vary by application. Current metrics, such as nuScenes mAP, might not fully align with the goals of anomaly detection. Additionally, incorporating advanced teacher-student frameworks in the semi-supervised setting could

lead to more sophisticated loss functions and improved model performance. Lastly, retraining the base detector using pseudo labels refined by a high-performing anomaly detector, or training an anomaly detector on its own filtered outputs, could open new areas for performance optimization.

By addressing these three domains, future work could further advance our work on anomaly detection in LiDAR-based object detection, pushing the boundaries of precision, efficiency, and scalability in real-world applications.

A. CenterPoint 100% seed results

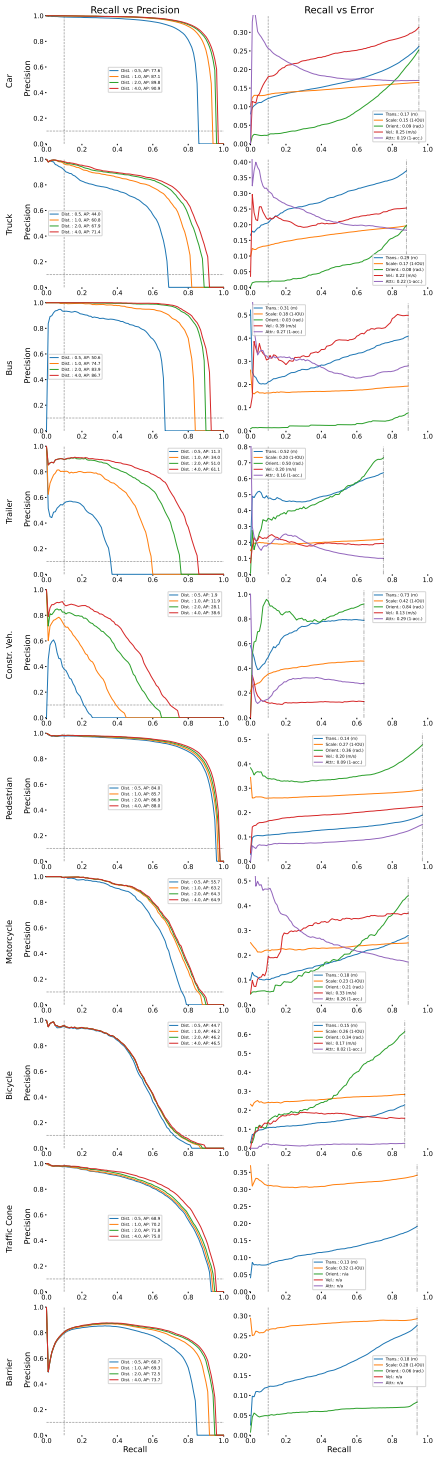


Figure A.1.: Results for the nuScenes' validation dataset from nuScenes-devkit evaluation tool for CenterPoint trained on 100% of nuScenes' training dataset.

B. CenterPoint 5% seed results

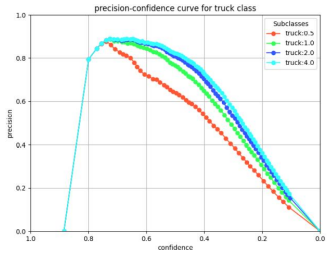
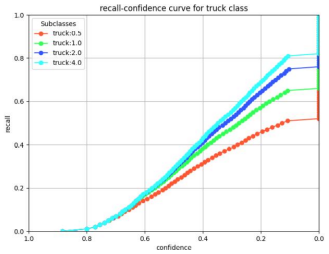
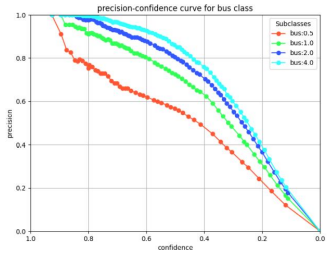
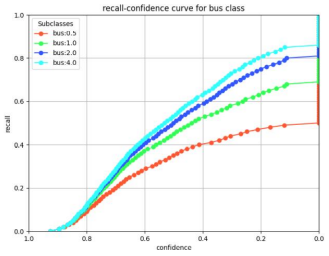
Precision-Confidence	Recall-Confidence
	
	

Table B.1.: Precision-Confidence and Recall-Confidence curves for the classes Truck and Bus for nuScene's validation dataset generated by CenterPoint, trained on 5% of nuScene's training dataset.

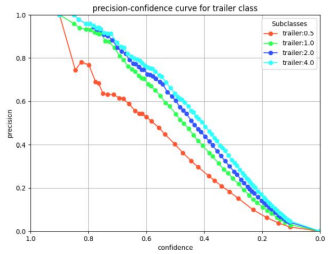
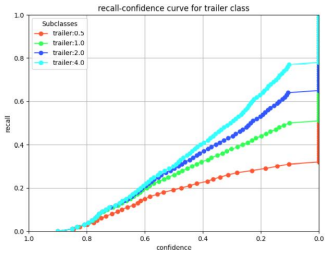
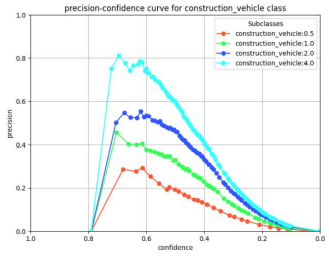
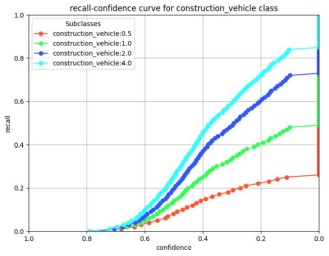
Precision-Confidence	Recall-Confidence
	
	

Table B.2.: Precision-Confidence and Recall-Confidence curves for the classes Trailer and Construction Vehicle for nuScene's validation dataset generated by CenterPoint, trained on 5% of nuScene's training dataset.

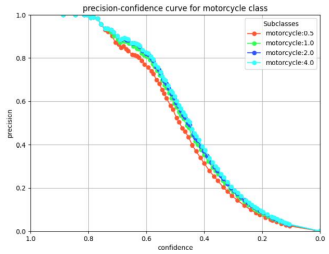
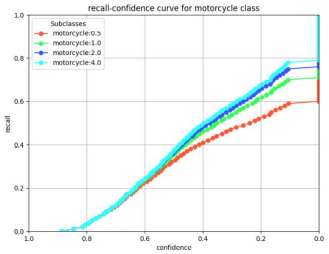
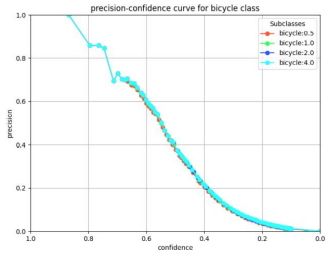
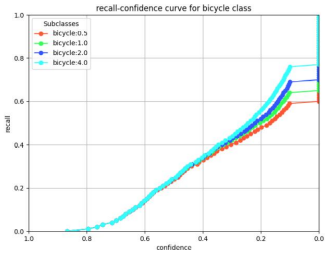
Precision-Confidence	Recall-Confidence
	
	

Table B.3.: Precision-Confidence and Recall-Confidence curves for the classes Motorcycle and Bicycle for nuScene's validation dataset generated by CenterPoint, trained on 5% of nuScene's training dataset.

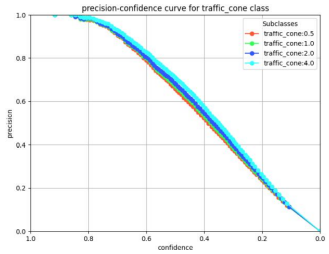
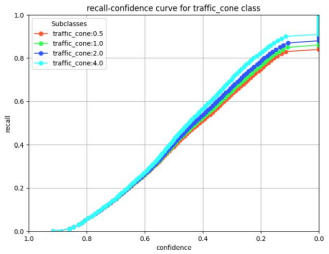
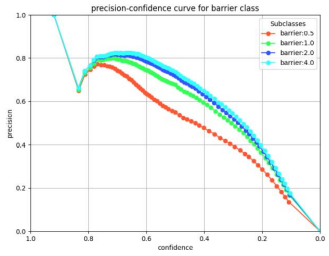
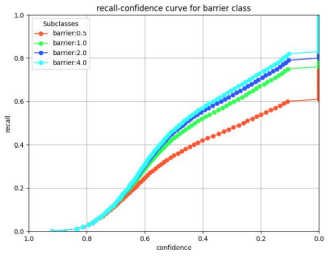
Precision-Confidence	Recall-Confidence
	
	

Table B.4.: Precision-Confidence and Recall-Confidence curves for the classes Traffic Cone and Barrier for nuScene's validation dataset generated by CenterPoint, trained on 5% of nuScene's training dataset.

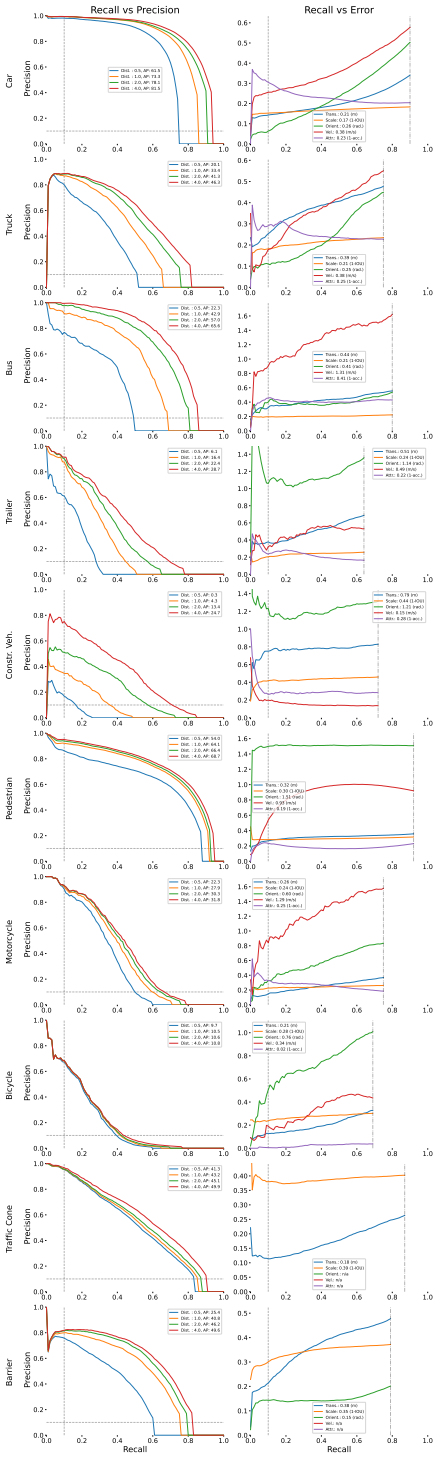


Figure B.1.: Results for the nuScenes' validation dataset from nuScenes-devkit evaluation tool for CenterPoint trained on 5% of nuScenes' training dataset.

C. Optuna Hyperparameter Study

Hyperparameter tuning for the model was conducted using Optuna [2], with the `MedianPruner` employed as the pruning strategy. This approach prunes unpromising trials based on their performance relative to the median of all trials, enabling efficient exploration of the hyperparameter space.

The Optuna study included 13 completed trials, each sampling hyperparameters listed in Table C.1. Model performance was measured based on accuracy on the development validation dataset (Section 5.2).

Hyperparameter	**Selection Range or Set**
Learning Rate	$[1e^{-5}, 5e^{-3}]$
Number of Multihead-Attention Heads (MHA)	$\{8, 16\}$
Number of Layers in LSTM	$\{3, 4, 5, 6\}$
Hidden Layer Size in LSTM	$[128, 512]$ (step size: 64)
Hidden Layer Size in MLP	$[128, 512]$ (step size: 64)

Table C.1.: AnomalyAttention's hyperparameter and their sampling ranges for Optuna study

The most important parameters identified by Optuna are shown in Figure C.1, with the final hyperparameters selected from the best-performing trial (Table C.2).

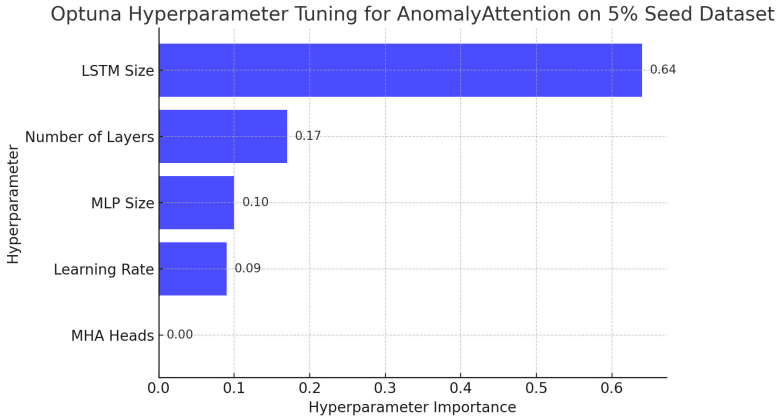


Figure C.1.: Hyperparameter by Importance based on Optuna Study for for AnomalyAttention, evaluated on 5% Seed Dataset

Table C.2.: Hyperparameter Tuning Results for AnomalyAttention on 5% Seed Dataset

Status	Accuracy	LSTM Size	# Layers	MLP Size	Learning Rate	# MHA Heads
COMPLETE	40.786328	256	6	256	0.000133	8
COMPLETE	39.307536	192	4	512	0.000198	16
COMPLETE	34.534667	384	5	192	0.000025	16
COMPLETE	15.956787	384	6	192	0.000138	16
COMPLETE	15.956787	128	5	512	0.000542	8
COMPLETE	15.956787	384	5	128	0.000397	16
COMPLETE	15.956787	256	5	512	0.000516	16
COMPLETE	15.956787	320	3	448	0.000778	8
COMPLETE	15.956787	448	3	128	0.000723	8
COMPLETE	15.956787	512	4	256	0.000290	16
COMPLETE	15.956787	128	6	384	0.000530	16
COMPLETE	15.956787	384	5	256	0.000419	16
COMPLETE	15.956787	448	6	320	0.000406	16
PRUNED	0.384418	512	4	512	0.000136	8
PRUNED	0.381963	320	5	448	0.000732	16
PRUNED	0.381187	512	6	192	0.000655	8
PRUNED	0.376211	512	3	128	0.000307	16
PRUNED	0.371874	384	5	448	0.000284	8
PRUNED	0.366542	448	4	192	0.000919	8
PRUNED	0.358888	256	4	128	0.000958	16
PRUNED	0.353432	192	3	512	0.000388	8
PRUNED	0.346661	256	4	128	0.000929	16
PRUNED	0.339812	192	4	320	0.000736	16
PRUNED	0.338574	320	6	512	0.000670	8

D. Qualitative Results

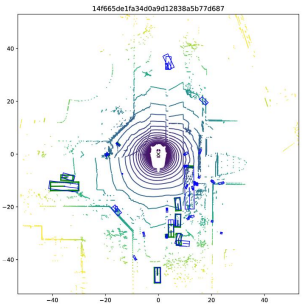
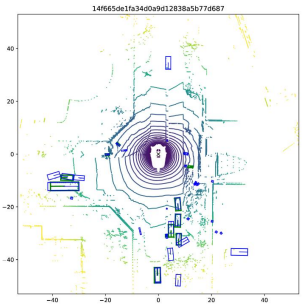
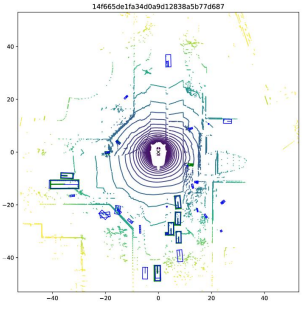
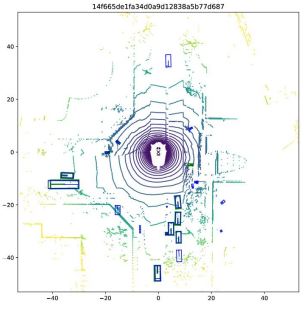
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.1.: Qualitative Results for nuScenes frame 14f665de1fa34d0a9d12838a5b77d687. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

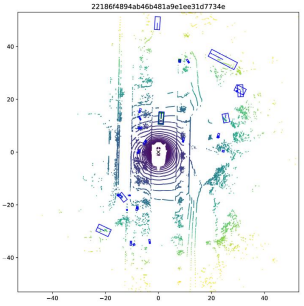
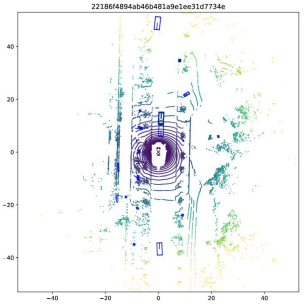
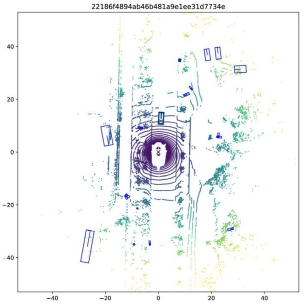
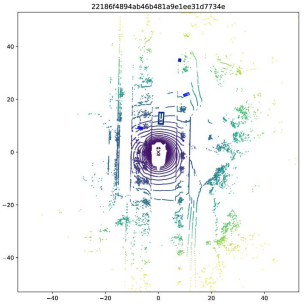
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.2.: Qualitative Results for nuScenes frame 22186f4894ab46b481a9e1ee31d7734e. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

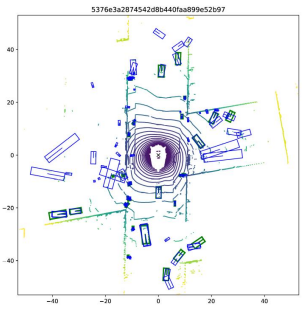
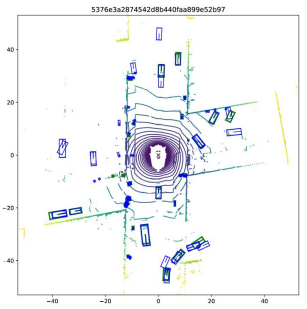
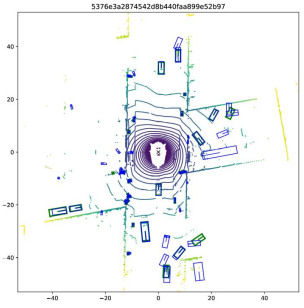
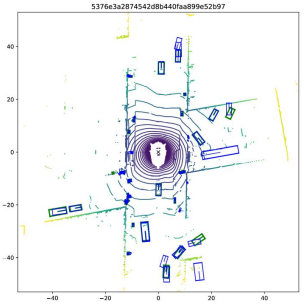
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.3.: Qualitative Results for nuScenes frame 5376e3a2874542d8b440faa899e52b97. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

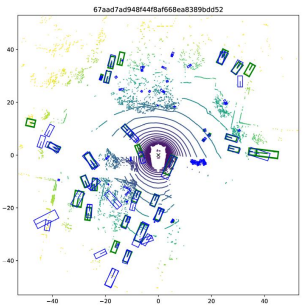
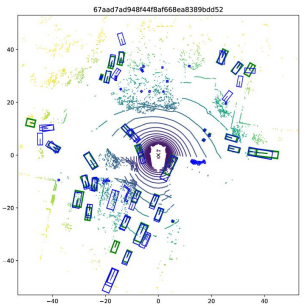
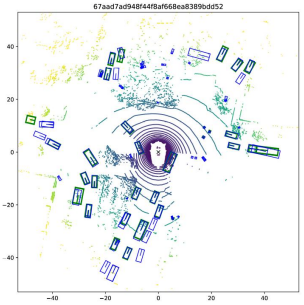
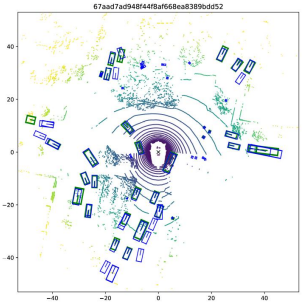
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.4.: Qualitative Results for nuScenes frame 67aad7ad948f44f8af668ea8389bdd52. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

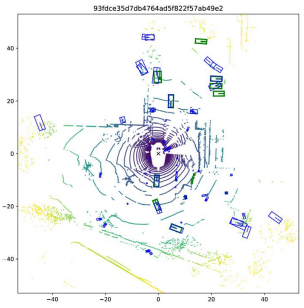
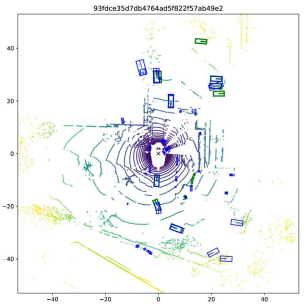
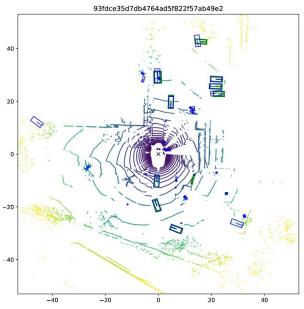
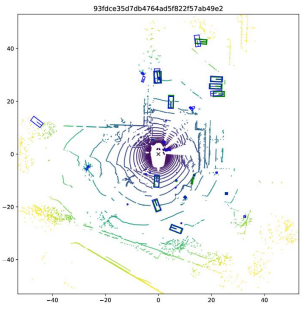
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.5.: Qualitative Results for nuScenes frame 93fdce35d7db4764ad5f822f57ab49e2. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

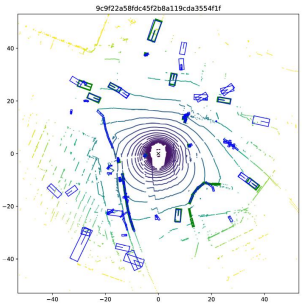
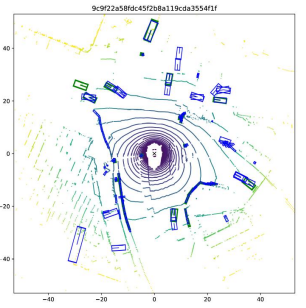
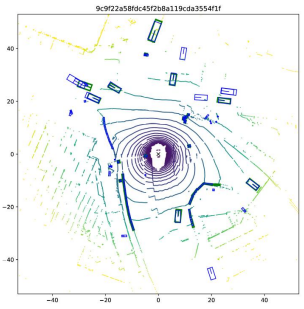
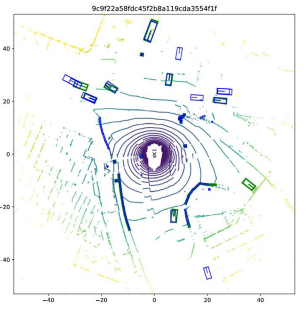
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.6.: Qualitative Results for nuScenes frame 9c9f22a58fdc45f2b8a119cda3554f1f. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

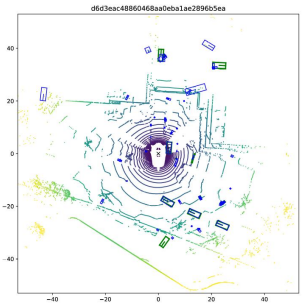
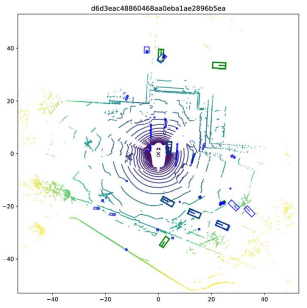
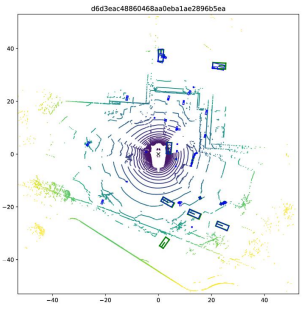
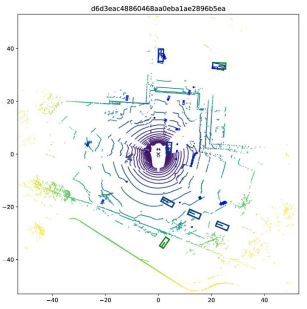
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.7.: Qualitative Results for nuScenes frame d6d3eac48860468aa0eba1ae2896b5ea. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

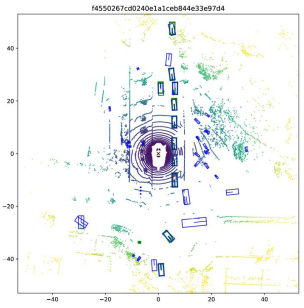
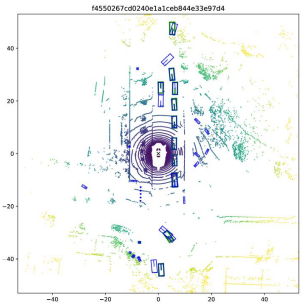
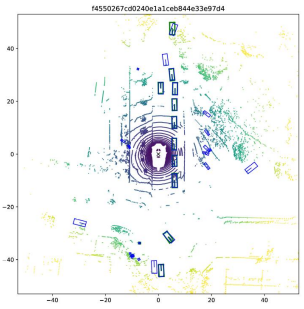
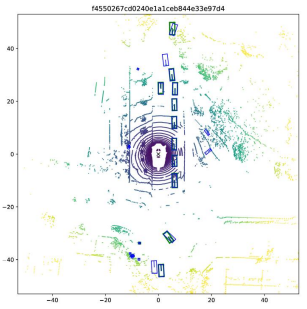
CenterPoint 5% Seed	CenterPoint+Ours 5% Seed
	
CenterPoint 100% Seed	**CenterPoint+Ours 100% Seed**
	

Table D.8.: Qualitative Results for nuScenes frame f4550267cd0240e1a1ceb844e33e97d4. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

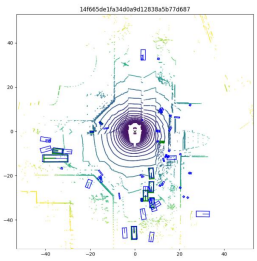
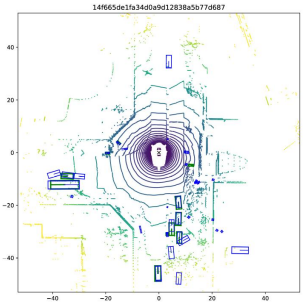
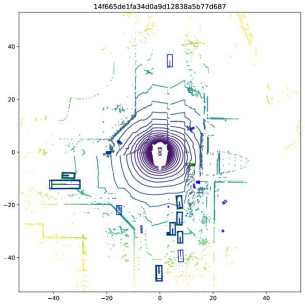
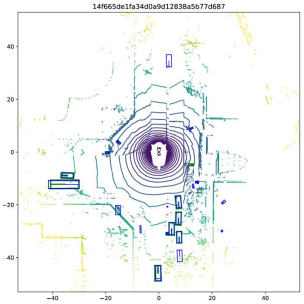
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.9.: Qualitative Results for nuScenes frame 14f665de1fa34d0a9d12838a5b77d687. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

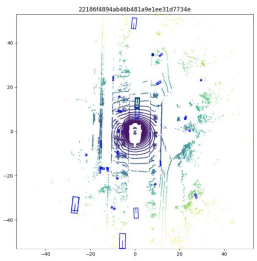
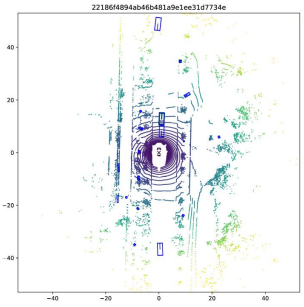
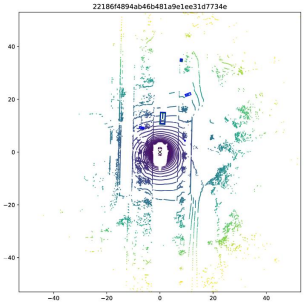
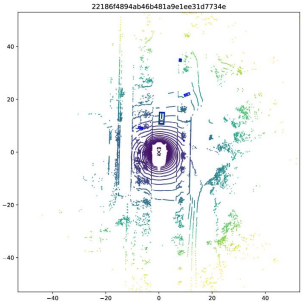
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.10.: Qualitative Results for nuScenes frame 22186f4894ab46b481a9e1ee31d7734e. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

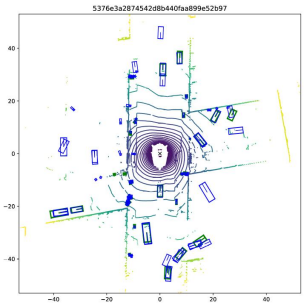
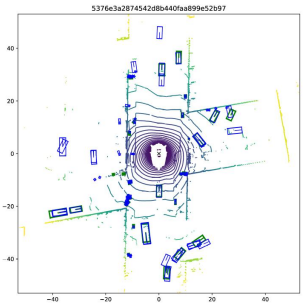
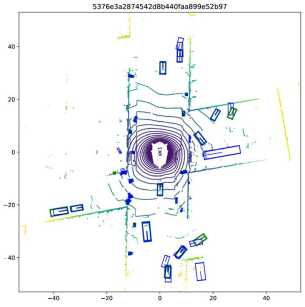
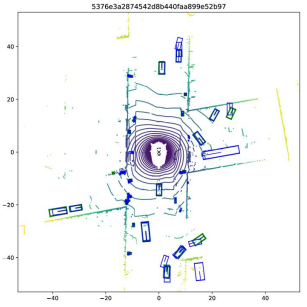
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.11.: Qualitative Results for nuScenes frame 5376e3a2874542d8b440faa899e52b97. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

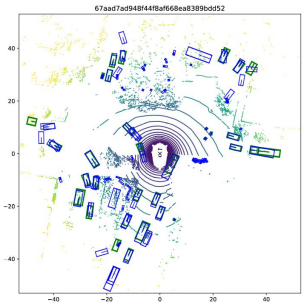
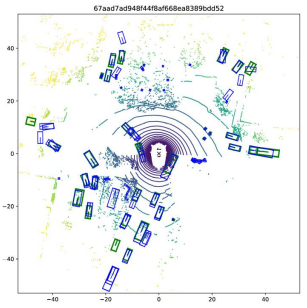
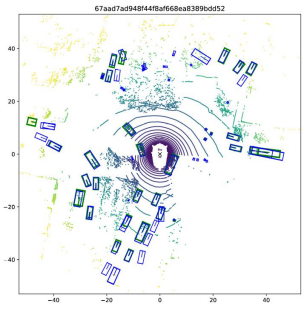
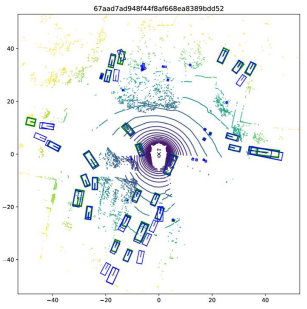
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.12.: Qualitative Results for nuScenes frame 67aad7ad948f44f8af668ea8389bdd52. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

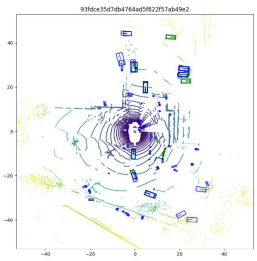
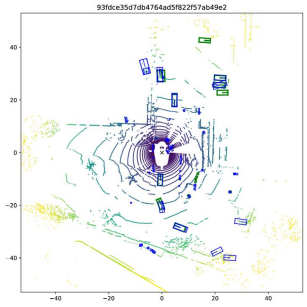
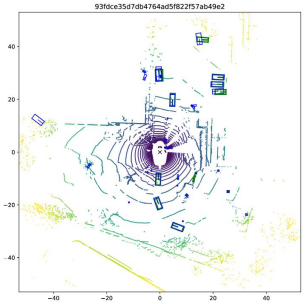
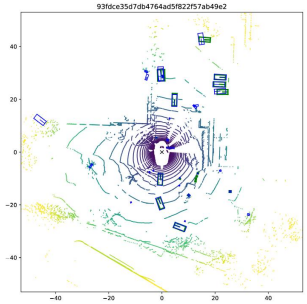
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.13.: Qualitative Results for nuScenes frame 93fdce35d7db4764ad5f822f57ab49e2 without AnomalyAttention compared to with AnomalyAttention. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

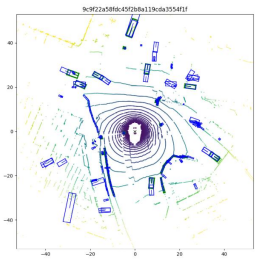
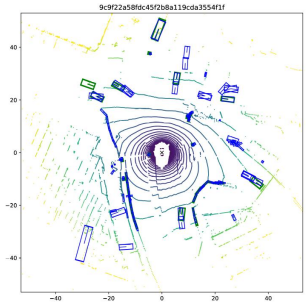
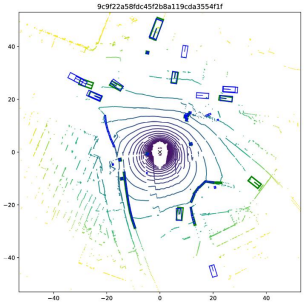
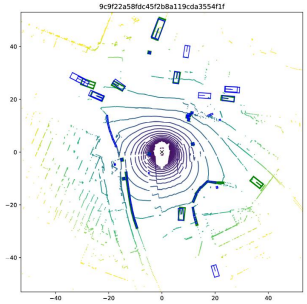
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.14.: Qualitative Results for nuScenes frame 9c9f22a58fdc45f2b8a119cda3554f1f without AnomalyAttention compared to with AnomalyAttention. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

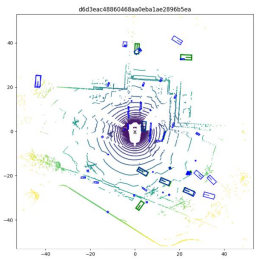
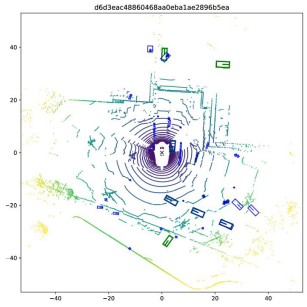
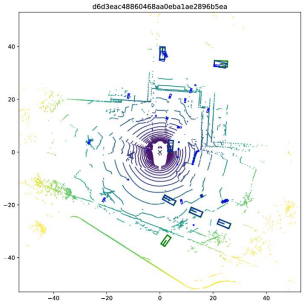
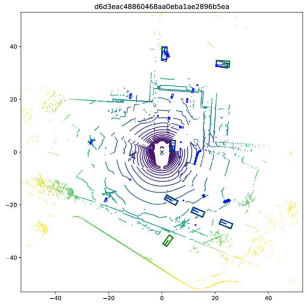
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.15.: Qualitative Results for nuScenes frame d6d3eac48860468aa0eba1ae2896b5ea without AnomalyAttention compared to with AnomalyAttention. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

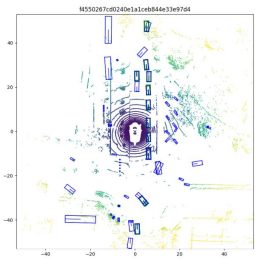
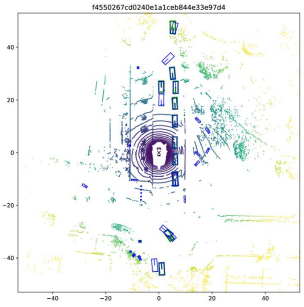
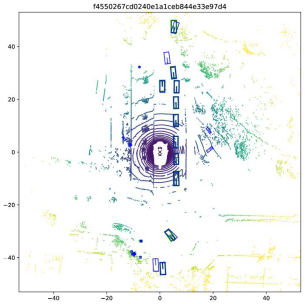
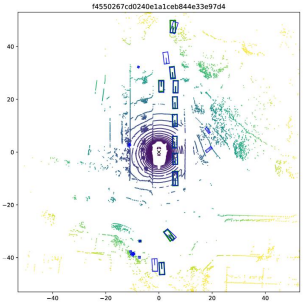
No AnomalyAtt. (5% Seed)	AnomalyAtt. (5% Seed)
	
No AnomalyAtt. (100% Seed)	**AnomalyAtt. (100% Seed)**
	

Table D.16.: Qualitative Results for nuScenes frame f4550267cd0240e1a1ceb844e33e97d4 without AnomalyAttention compared to with AnomalyAttention. Predicted bounding boxes are (blue), and ground truth bounding boxes are (green).

List of Figures

List of Tables

Bibliography

[1] Scale AI. *Scale AI Official Website*. Accessed: 09/12/2024. URL: https://scale.com/.

[2] Takuya Akiba et al. "Optuna: A Next-Generation Hyperparameter Optimization Framework." In: *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2623–2631.

[3] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. "People-tracking-by-detection and people-detection-by-tracking." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587583.

[4] Shai Avidan. "Ensemble Tracking." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2 (2007), pp. 261–271. DOI: 10.1109/TPAMI.2007.35.

[5] Jürgen Bernard et al. "Comparing Visual-Interactive Labeling with Active Learning: An Experimental Study." In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 298–308. DOI: 10.1109/TVCG.2017.2744818.

[6] Holger Caesar et al. "nuScenes: A multimodal dataset for autonomous driving." In: *CoRR* abs/1903.11027 (2019). arXiv: 1903.11027. URL: http://arxiv.org/abs/1903.11027.

[7] Liangyu Chen et al. "Points as Queries: Weakly Semi-supervised Object Detection by Points." In: *CoRR* abs/2104.07434 (2021). arXiv: 2104.07434. URL: https://arxiv.org/abs/2104.07434.

[8] Zeming Chen et al. *Mixed Pseudo Labels for Semi-Supervised Object Detection*. 2023. arXiv: 2312.07006 [cs.CV]. URL: https://arxiv.org/abs/2312.07006.

[9] CVAT.ai Corporation. *Computer Vision Annotation Tool (CVAT)*. Version 2.8.2. Nov. 2023. URL: https://github.com/cvat-ai/cvat.

[10] *CTRL Instructions - Vehicle Class*. Online. [Accessed January 21, 2025]. URL: https://github.com/tusen-ai/SST/blob/main/docs/CTRL_instructions.md.

[11] M. Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge." In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[12] Lue Fan et al. *Once Detected, Never Lost: Surpassing Human Performance in Offline LiDAR based 3D Object Detection*. 2023. arXiv: 2304.12315 [cs.CV]. URL: https://arxiv.org/abs/2304.12315.

[13] Xin Fan et al. "An interactive visual analytics approach for network anomaly detection through smart labeling." In: *Journal of Visualization* 22 (Sept. 2019). DOI: 10.1007/s12650-019-00580-7.

[14] Whye Kit Fong et al. "Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking." In: *arXiv preprint arXiv:2109.03805* (2021).

[15] LF AI Data Foundation. *Xtreme1 - The Next GEN Platform For Multisensory Training Data*. Software available from https://github.com/xtreme1-io/xtreme1/. 2023. URL: https://xtreme1.io/.

[16] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

[17] Zhihao Gu et al. "Remembering Normality: Memory-guided Knowledge Distillation for Unsupervised Anomaly Detection." In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 16355–16363. DOI: 10.1109/ICCV51070.2023.01503.

[18] Hewei Guo et al. " Template-guided Hierarchical Feature Restoration for Anomaly Detection." In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct. 2023, pp. 6424–6435. DOI: 10.1109/ICCV51070.2023.00593. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.00593.

[19] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[20] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV]. URL: https://arxiv.org/abs/1502.01852.

[21] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[22] Motional Holger Caesar. *nuScnees annotation tool*. Accessed: 09/12/2024. 2019. URL: https://github.com/nutonomy/nuScenes-devkit/issues/244.

[23] Eliahu Horwitz and Yedid Hoshen. *Back to the Feature: Classical 3D Features are (Almost) All You Need for 3D Anomaly Detection*. 2022. arXiv: 2203.05550 [cs.CV]. URL: https://arxiv.org/abs/2203.05550.

[24] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems." In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. DOI: 10.1115/1.3662552.

[25] Nitin Kumar Saravana Kannan, Matthias Reuse, and Martin Simon. "Click Crop & Detect: One-Click Offline Annotation for Human-in-the-Loop 3D Object Detection on Point Clouds." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2024, pp. 4514–4525.

[26] Gary King and Langche Zeng. "Logistic Regression in Rare Events Data." In: *Political Analysis* 9.2 (2001), pp. 137–163. DOI: 10.1093/oxfordjournals.pan.a004868.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[28] Harold W. Kuhn. "The Hungarian method for the assignment problem." In: *Naval Research Logistics Quarterly* 2.1–2 (1955), pp. 83–97. DOI: 10.1002/nav.3800020109.

[29] Alex H. Lang et al. "PointPillars: Fast Encoders for Object Detection from Point Clouds." In: *CoRR* abs/1812.05784 (2018). arXiv: 1812.05784. URL: http://arxiv.org/abs/1812.05784.

[30] Alexander Lehner et al. "3D-VField: Learning to Adversarially Deform Point Clouds for Robust 3D Object Detection." In: *CoRR* abs/2112.04764 (2021). arXiv: 2112.04764. URL: https://arxiv.org/abs/2112.04764.

[31] Qing Lian et al. "Semi-supervised Monocular 3D Object Detection by Multi-view Consistency." In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 715–731.

[32] Hanxue Liang et al. "Exploring Geometry-aware Contrast and Clustering Harmonization for Self-supervised 3D Object Detection." In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 3273–3282. DOI: 10.1109/ICCV48922.2021.00328.

[33] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection." In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[34] Jiaqi Liu et al. *Real3D-AD: A Dataset of Point Cloud Anomaly Detection*. 2023. arXiv: 2309.13226 [cs.CV]. URL: https://arxiv.org/abs/2309.13226.

[35] Tao Ma et al. "DetZero: Rethinking Offboard 3D Object Detection with Long-term Sequential Point Clouds." In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 6713–6724. DOI: 10.1109/ICCV51070.2023.00620.

[36] Motional. *Motional Official Website*. https://motional.com/nuScenes. Accessed: 09/12/2024.

[37] Jinhyung Park et al. *DetMatch: Two Teachers are Better Than One for Joint 2D and 3D Semi-Supervised Object Detection*. 2022. arXiv: 2203.09510 [cs.CV]. URL: https://arxiv.org/abs/2203.09510.

[38] Jonah Philion, Amlan Kar, and Sanja Fidler. "Learning to Evaluate Perception Models Using Planner-Centric Metrics." In: *CoRR* abs/2004.08745 (2020). arXiv: 2004.08745. URL: https://arxiv.org/abs/2004.08745.

[39] *PyTorch Documentation: BCEWithLogitsLoss*. Online. [Accessed January 4, 2025]. URL: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html#torch.nn.BCEWithLogitsLoss.

[40] *PyTorch Documentation: LSTM*. Online. [Accessed January 4, 2025]. URL: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html.

[41] *PyTorch Documentation: Transformer*. Online. [Accessed January 4, 2025]. URL: https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html.

[42] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV]. URL: https://arxiv.org/abs/1612.00593.

[43] Sebastian Raschka. "An Overview of General Performance Metrics of Binary Classifier Systems." In: *CoRR* abs/1410.5330 (2014). arXiv: 1410.5330. URL: http://arxiv.org/abs/1410.5330.

[44] Dimensional Research. *Artificial Intelligence and Machine Learning Projects Are Obstructed by Data Issues*. https://cdn2.hubspot.net/hubfs/3971219/Survey%20Assets%201905Dimensional%20Research%20Machine%20Learning%20PPT%20Report%20FINAL.pdf. Accessed: 09/12/2024. 2019.

[45] Marco Rudolph et al. *Asymmetric Student-Teacher Networks for Industrial Anomaly Detection*. 2022. arXiv: 2210.07829 [cs.LG]. URL: https://arxiv.org/abs/2210.07829.

[46] *sklearn.utils.class$_w$eight.compute$_c$lass$_w$eight*. Online. [Accessed January 4, 2025]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html.

[47] Waqas Sultani, Chen Chen, and Mubarak Shah. "Real-world Anomaly Detection in Surveillance Videos." In: *CoRR* abs/1801.04264 (2018). arXiv: 1801.04264. URL: http://arxiv.org/abs/1801.04264.

[48] Pei Sun et al. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset." In: *CoRR* abs/1912.04838 (2019). arXiv: 1912.04838. URL: http://arxiv.org/abs/1912.04838.

[49] Antti Tarvainen and Harri Valpola. "Weight-averaged consistency targets improve semi-supervised deep learning results." In: *CoRR* abs/1703.01780 (2017). arXiv: 1703.01780. URL: http://arxiv.org/abs/1703.01780.

[50] Yizhak Vaisman et al. *Detecting Anomalous Network Communication Patterns Using Graph Convolutional Networks*. 2023. arXiv: 2311.18525 [cs.CR]. URL: https://arxiv.org/abs/2311.18525.

[51] Ashish Vaswani et al. "Attention Is All You Need." In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[52] ITRexGroup Victoria Shashkina. *Calculating machine learning costs: price factors and estimates from the ITRex portfolio*. https://itrexgroup.com/blog/machine-learning-costs-price-factors-and-estimates/. Accessed: 09/12/2024. 2024.

[53] He Wang et al. *3DIoUMatch: Leveraging IoU Prediction for Semi-Supervised 3D Object Detection*. 2021. arXiv: 2012.04355 [cs.CV]. URL: https://arxiv.org/abs/2012.04355.

[54] Jianren Wang et al. "Semi-supervised 3D Object Detection via Temporal Graph Neural Networks." In: *CoRR* abs/2202.00182 (2022). arXiv: 2202.00182. URL: https://arxiv.org/abs/2202.00182.

[55] Qitai Wang et al. "Immortal Tracker: Tracklet Never Dies." In: *CoRR* abs/2111.13672 (2021). arXiv: 2111.13672. URL: https://arxiv.org/abs/2111.13672.

[56] Waymo. *Waymo Official Website*. https://waymo.com. Accessed: 09/12/2024.

[57] Xinshuo Weng and Kris Kitani. "A Baseline for 3D Multi-Object Tracking." In: *CoRR* abs/1907.03961 (2019). arXiv: 1907.03961. URL: http://arxiv.org/abs/1907.03961.

[58] Xingjiao Wu et al. "A Survey of Human-in-the-loop for Machine Learning." In: *CoRR* abs/2108.00941 (2021). arXiv: 2108.00941. URL: https://arxiv.org/abs/2108.00941.

[59] Cheng Yan et al. "Feature Prediction Diffusion Model for Video Anomaly Detection." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 5527–5537.

[60] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. "Center-based 3D Object Detection and Tracking." In: *CoRR* abs/2006.11275 (2020). arXiv: 2006.11275. URL: https://arxiv.org/abs/2006.11275.

[61] Sergey Zakharov et al. "Autolabeling 3D Objects with Differentiable Rendering of SDF Shape Priors." In: *CoRR* abs/1911.11288 (2019). arXiv: 1911.11288. URL: http://arxiv.org/abs/1911.11288.

[62] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. "SESS: Self-Ensembling Semi-Supervised 3D Object Detection." In: *CoRR* abs/1912.11803 (2019). arXiv: 1912.11803. URL: http://arxiv.org/abs/1912.11803.

[63] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection." In: *CoRR* abs/1711.06396 (2017). arXiv: 1711.06396. URL: http://arxiv.org/abs/1711.06396.

[64] Qinfeng Zhu, Lei Fan, and Ningxin Weng. "Advancements in point cloud data augmentation for deep learning: A survey." In: *Pattern Recognition* 153 (Sept. 2024), p. 110532. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2024.110532. URL: http://dx.doi.org/10.1016/j.patcog.2024.110532.