Towards Object Re-identification from Point Clouds for 3D MOT

by

Benjamin Thérien

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

Statement of Contributions

This thesis is based on a research paper that is currently in submission involving joint work with Chengjie Huang, Adrian Chow, and Dr. Krzysztof Czarnecki. Chengjie Huang's contributions were twofold: 1) he wrote a library for aggregating point cloud observations from the WOD and nuScenes dataset, which I built upon, adding the ability to extract objects from a 3D object detector's predicted bounding boxes and to project and extract corresponding image crops, to create the ReID datasets and 2) he trained the object detector on WOD, provided corresponding detections, and helped to write the related sections in the paper. Adrian Chow was responsible for writing code to extract a tracker's errors from nuScenes track prediction files. Finally, Dr. Krzysztof Czarnecki made numerous edits to the main manuscript and advised me on the framing of our research paper.

Abstract

This thesis studies the problem of object re-identification (ReID) in a 3D multi-object tracking (MOT) context, by learning to match pairs of objects from cropped (e.g., using their predicted 3D bounding boxes) point cloud observations. We are not concerned with state-of-the-art performance for 3D MOT, however. Instead, we seek to answer the following question: In a realistic tracking by-detection context, how does object ReID from point clouds perform relative to ReID from images? To enable such a study, we propose a lightweight matching head that can be concatenated to any set or sequence processing backbone (e.g., PointNet or ViT), creating a family of comparable object ReID networks for both modalities. Run in Siamese style, our proposed point cloud ReID networks can make thousands of pairwise comparisons in real-time (10 Hz). Our findings demonstrate that their performance increases with higher sensor resolution and approaches that of image ReID when observations are sufficiently dense. Additionally, we investigate our network's ability to enhance 3D multi-object tracking, showing that our point cloud ReID networks can successfully re-identify objects that led a strong motion-based tracker into error. To our knowledge, we are the first to study real-time object re-identification from point clouds in a 3D multi-object tracking context.

Acknowledgements

I would first like to thank my advisor, Dr. Krzysztof Czarnecki for his belief in me and his willingness to discuss research ideas or questions whenever they occurred to me. You have inspired me to be a better researcher. I would also like to thank Dr. Michal Antkiewicz for his willingness to troubleshoot technical issues with me and his role in enhancing our computing environment.

I would like to acknowledge my research collaborators. Particularly Chengjie Huang and Adrian Chow for their contributions to our research paper (in submission). Chengjie Huang for his endless willingness to troubleshoot technical issues or discuss research ideas. Finally, I would like to thank Luke Rowe for the many enlightening discussions we had in our shared office space.

Dedication

This thesis is dedicated to my parents, Carol and Claude, sister, Laura, girlfriend, Ria, and my friends, too many to name here—you know who you are.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acc.	Accuracy
BCE	Binary Cross-entropy
BEV	Bird's Eye View
CFA	Cross-feature Augmentation
C+L/CL	Camera and LiDAR
DGCNN	Deep Graph Convolutional Neural Network
DeiT	Data-efficient Image Transformer
FP	False Positive
FOV	Field of View
GB	Gigabyte
GPU	Graphics Processing Unit
GT	Ground Truth
IoU	Intersection over union
LCA	Linear Cross Attention
LiDAR	Light Detection and Ranging
LN	Layer Normalization
LSTM	Long Short-term Memory
MHA	Multi-heat Attention
MLE	Maximum likelihood estimator
MLP	Multilayer perceptron
ms	millisecond
MOT	multi-object tracking
Neg.	Negative
NLP	Natural Language Processing
PT	Pre-trained
Pos.	Positive
ReID	re-identification
RTMM	real-time matching module
SE(3)	Special Euclidean Group
SO(3)	Special Orthogonal Group
SOT	single-object tracking
TP	True Positive
ViT	Vision Transformer
WOD	Waymo Open Dataset

List of Symbols

Our mathematical notation was inspired by that used in [36].

$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix or tensor of specified dimension
$\boldsymbol{A}_{i,:}$	ith row of a matrix
$\boldsymbol{A}_{:,j}$	jth column of a matrix
$\mathbb{R}$	The real numbers
f_θ	A function parameterized by θ
x^+	$\max(x, 0)$
$\sigma(x)$	Sigmoid activation function, $\frac{1}{1+e^{-x}}$
$\nabla_\theta x$	Gradient of x with respect to θ
$\mathcal{P}(\boldsymbol{x})$	Distribution of $\boldsymbol{x}$
$\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$	Joint distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$
$\mathcal{P}(\boldsymbol{y}\vert\boldsymbol{x})$	Conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$
$x \sim \mathcal{P}$	x has distribution $\mathcal{P}$
$\mathbb{E}_{\boldsymbol{y}\sim\mathcal{P}\vert\boldsymbol{x}} f(\boldsymbol{y})$	Conditional expectation of a function f of a random variable $\boldsymbol{y}$ given $\boldsymbol{x}$

Chapter 1

Introduction

Multi-object tracking is essential for safety-critical autonomous driving, where being able to accurately keep state can be a matter of life or death. Many 3D MOT methods [155, 75, 160, 57] following the tracking-by-detection paradigm rely heavily on motion cues provided by the precise detections of LiDAR-based methods [155, 75]. While these trackers easily arrive close to the top of existing benchmarks, they do not capture the complete picture. Any complete multi-object tracking approach should feature a re-identification component in addition to a motion-based tracker, especially for safety-critical applications. Consider, for instance, Figure 1.1, where the Ego vehicle, E, equipped with only a motion-based tracker, loses sight of V1 due to occlusion and incorrectly propagates V1's track to V2. This causes a potential crash scenario in frame 6, where the ego is trying to reach the exit (unaware of V1), while V1 is slowing down to pass behind the truck. Had the ego been equipped with a ReID module, the motion model's error in frame 3 could have been avoided. Therein lies the importance of object ReID for 3D MOT—it is robust to ambiguous motion cues which can fool even the strongest motion-based trackers. Consequently, many existing works leverage ReID features from images to improve a motion-based tracker's performance [58, 142, 141, 133].

While strong ReID performance can be obtained from RGB data alone, even autonomous agents equipped with arrays of RGB sensors stand to benefit from the added redundancy, diversity, and complementarity offered by processing depth-sensor information for ReID. Let us examine, for instance, an autonomous vehicle equipped with both types of sensors. Redundancy refers to the ability of one sensor to accomplish the job of another if it were to fail. If the camera malfunctions, then LiDAR can fill the gap. Diversity refers to the range of information available from different sensors. When added to RGB information, LiDAR scans offer otherwise unavailable 3D shape information, increasing the diversity of the sensor suite. Complementarity refers to the ability of two sensors to complement each other. When processing LiDAR scans in addition to image information, the sensors complement each other by having slightly different viewing angles and being robust to different lighting conditions. Together with the need for ReID in a tracking context, this requirement for independently leveraging point cloud information in multi-modal settings leads us to wonder: *how effective is LiDAR-based ReID compared to camera-based ReID in a tracking-by-detection scenario?*

To the best of our knowledge, we are the first to investigate ReID of deformable and non-deformable objects from sparse LiDAR observations in a tracking-by-detection context. Our contributions can be summarized as follows:

- We propose a symmetric matching head that runs in real-time and can be added to most existing point processing backbones.
- Our thorough empirical evaluation on the Waymo and nuScenes datasets is the first to establish point cloud ReID performance relative to image ReID. Our results demonstrate that our proposed point cloud ReID networks can approach the performance of image ReID when LiDAR observations are sufficiently dense.
- Our experiments investigating the effect of scaling training time reveal that our networks continue to learn and improve well beyond their original training schedule, suggesting a simple way to improve performance.
- We demonstrate that our networks can successfully re-identify observations that led three strong motion-based multi-object trackers into error.

Our results outline a promising future for point-based object ReID, especially as depth-sensor resolution continues to increase.

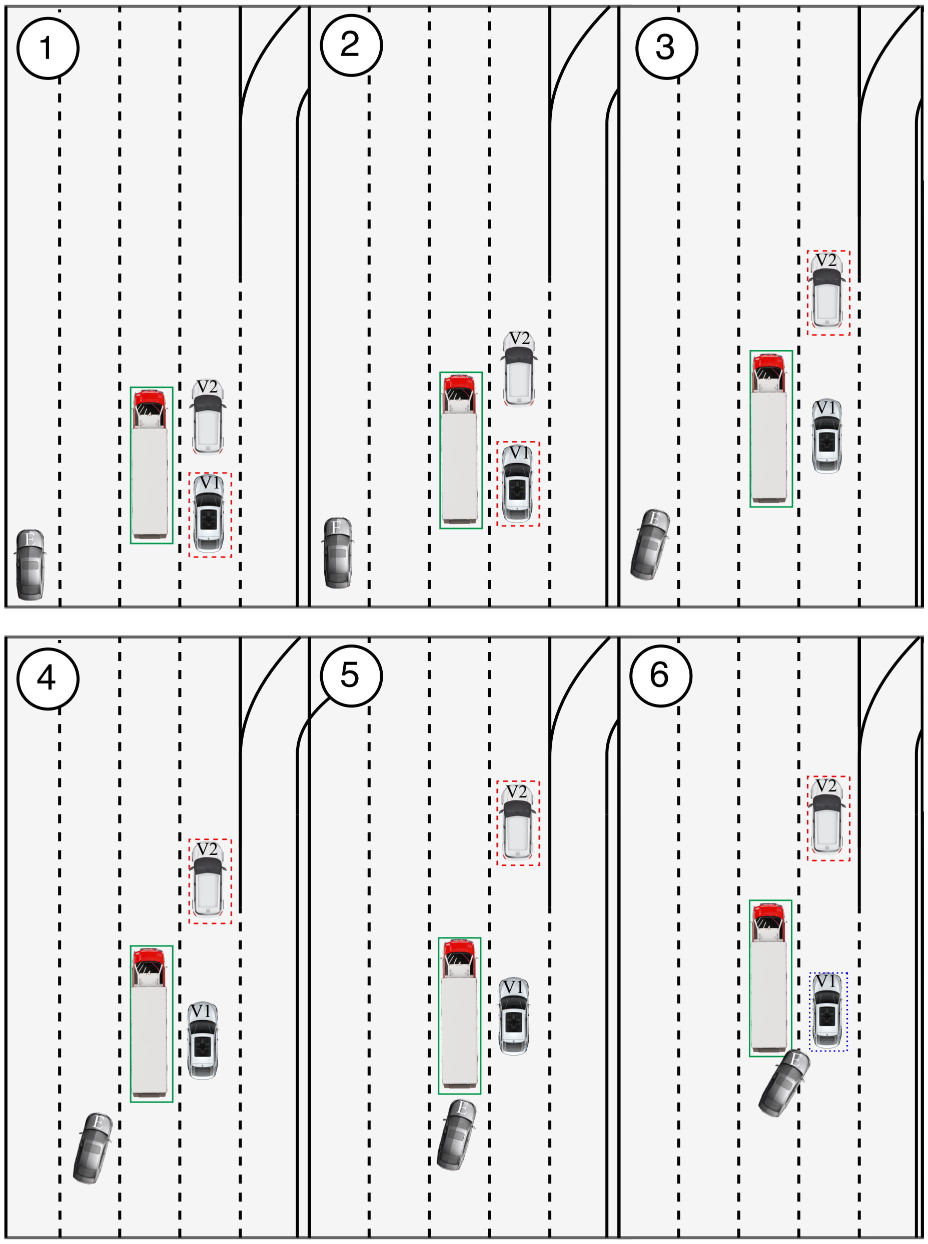

Figure 1.1: **Only relying on motion cues for MOT can be problematic.** The sequence depicts a potential crash scenario that could have been avoided by a tracker leveraging object re-identification.

Chapter 2

Background

We begin this chapter with a brief overview of supervised learning and its connection to maximum likelihood estimation. We then explain how this powerful methodology can be applied to learn representations of complex high-dimensional data, a necessary step to effectively re-identify objects from sensor observations. We continue our discussion of deep learning by introducing the transformer, an architecture quickly becoming ubiquitous to most applications, and show how it can be adapted to process images and point clouds.

2.1 Supervised Learning

Let $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$ be a probability distribution relating $\boldsymbol{x}$ and $\boldsymbol{y}$. Supervised learning algorithms relate $\boldsymbol{x}$ to $\boldsymbol{y}$ by making a *prediction* of $\boldsymbol{y}$'s value given an observation of $\boldsymbol{x}$. Suppose that $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^k$ and $\boldsymbol{y} \in Y \subseteq \mathbb{R}^g$, then the goal of supervised learning is to find a function $f : \mathcal{X} \to Y$ which estimates the conditional density $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$. This differs from unsupervised learning, which is typically concerned with directly learning the density $\mathcal{P}(\boldsymbol{x})$ or clustering observations $\boldsymbol{x} \sim \mathcal{P}(\boldsymbol{x})$ without requiring access to the corresponding label $\boldsymbol{y}$.

While non-parametric methods to supervised learning exist (e.g. k-Nearest-Neighbour), in what follows we will concern ourselves exclusively with methods that are parametric. Formally, supervised learning can be defined as the problem of estimating the parameters θ of a function $f_\theta \in \mathcal{F}$ that minimizes the expected prediction error over $\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$, where $\mathcal{F}$ is a class of functions mapping $\mathcal{X}$ to Y. The expected prediction error is defined as the expected loss over the distribution $\mathcal{P}$:

$$\mathcal{L}(f_\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{P}}[l(f_\theta(\boldsymbol{x}), \boldsymbol{y})], \tag{2.1}$$

where $l : Y \times Y \to \mathbb{R}$ is a loss function that measures the discrepancy between the predicted output $f_\theta(\boldsymbol{x})$ and the true output $\boldsymbol{y}$. The function f that minimizes the expected prediction error is called the optimal predictor or the Bayes predictor, and it is defined as:

$$f^*(\boldsymbol{x}) = \arg\min_{\hat{\boldsymbol{y}} \in Y} \mathbb{E}_{\boldsymbol{y} \sim \mathcal{P}|\boldsymbol{x}}[l(\hat{\boldsymbol{y}}, \boldsymbol{y})], \tag{2.2}$$

where $\mathbb{E}_{\boldsymbol{y}' \sim \mathcal{P}|\boldsymbol{x}}$ denotes the conditional expectation of $\boldsymbol{y}$ given $\boldsymbol{x}$.

In practise, the true joint density $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$ is unknown and we only have access to a finite number of samples $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$. Therefore, finding the optimal predictor f^* is impossible. Instead, our supervised learning algorithms estimate the conditional density $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$ based on a finite dataset $\mathcal{D} =: \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ by finding a function $f_\theta \in \mathcal{F}$ that minimizes the empirical risk, defined as:

$$\hat{\mathcal{L}}(f_\theta) = \frac{1}{n} \sum_{i=1}^{n} l(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i). \tag{2.3}$$

This is equivalent to solving the following optimization problem:

$$\hat{\theta} = \arg\min_{\theta} \hat{\mathcal{L}}(f_\theta). \tag{2.4}$$

This optimization problem can be solved using various techniques, such as gradient descent, stochastic gradient descent, or convex optimization.

Under the assumption that $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are i.i.d and with the appropriate choice of loss function $\hat{\mathcal{L}}$ (e.g., cross-entropy), the empirical risk $\hat{\mathcal{L}}(f_\theta)$ is equivalent to the negative log-likelihood of the model parameters. Specifically, if we assume that the conditional distribution $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$ is approximated by a probability density function $f_\theta(\boldsymbol{y}|\boldsymbol{x})$, where θ is the parameter vector, then the negative log-likelihood (NLL) of the parameters under the training examples in $\mathcal{D}$ is given by:

$$NLL(\theta) = -\sum_{i=1}^{n} \log f_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i). \tag{2.5}$$

In this case, maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood, which is equivalent to minimizing the empirical risk $\hat{\mathcal{L}}(f)$, where $\hat{\mathcal{L}}$ is chosen to be cross-entropy. Therefore, $\hat{\theta}$ obtained from solving equation 2.4 will be the conditional maximum likelihood estimator (MLE) of $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$. Using MLE estimators is appealing in practice as they can be shown to be the best estimators in the limit as the number of samples $n \to \infty$ [36]; that is, they converge the fastest to the true value of θ. This, of course, assumes 1) that the true functional form of the conditional distribution $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$ is an element of $\mathcal{F}$ and can therefore be found by solving equation 2.4 and 2) that the true distribution has a single valid parametric form f_θ.

2.2 Representation Learning

The most popular supervised learning algorithms today fall under the category of Deep Learning. Unlike traditional supervised learning algorithms, which typically rely on models of limited representational capacity operating over handcrafted features, deep learning automatically extracts features of interest directly from the data. By stacking multiple layers of non-linear transformations, deep learning models can learn hierarchical representations that capture increasingly abstract and complex features. These high-capacity models can be used to directly estimate $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$ even when $\boldsymbol{x}$ is very high-dimensional, as is the case for images or point clouds.

Deep learning solutions to machine learning problems can be described by five main components: the dataset, loss function, model architecture, optimization algorithm, and hyperparameters. We will now describe each of these components by illustrating how they combine to solve a simple binary classification problem. Suppose that we are interested in diagnosing a disease from chest x-ray scans. We have access to a dataset $\mathcal{D} =: \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$, whose samples are drawn i.i.d from $\mathcal{P}(\boldsymbol{x}, \boldsymbol{y})$. $\boldsymbol{x} \in \mathcal{X}$ is a chest x-ray scan (2D image) that we have flattened into one long row vector. $\boldsymbol{y} \in \{0, 1\}$ is the corresponding label; $\boldsymbol{y} = 1$ if the patient has the disease and $\boldsymbol{y} = 0$ otherwise. Binary cross-entropy is a common loss function to use in this setting

$$\mathcal{L}_{bce}(x, y) = y \cdot \log(x) + (1 - y) \cdot \log(1 - x). \tag{2.6}$$

For illustration purposes, we select our model to be a simple multi-layer perceptron $f_\theta(\boldsymbol{x}) = \sigma(\boldsymbol{W}_h \cdot (\boldsymbol{W}_{h-1} \cdot \ldots (\boldsymbol{W}_2 \cdot (\boldsymbol{W}_1 \boldsymbol{x}^T)^+)^+))$ with h layers and ReLU non-linearities after each layer except for layer h whose activation function is a sigmoid. Bias terms can be included in $\boldsymbol{W}$ by appending a constant element to the input. The parameters θ are the weights of each matrix $\boldsymbol{W}_i, i = 1, 2, \ldots, h$. Having specified our dataset, loss function, and model architecture, we must now select an optimization algorithm to solve the optimization problem of equation 2.4. In deep learning, this is typically accomplished using gradient descent, stochastic gradient descent, or one of their variants [77, 59, 109]. In stochastic gradient descent, the parameters of f_θ are updated as follows

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \mathcal{L}_{bce}(f_\theta(\boldsymbol{x}), \boldsymbol{y}), \tag{2.7}$$

using the gradient of the loss function with respect to a single example. A more efficient variant called mini-batch gradient descent averages the gradient over a number of samples

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{bce}(f_\theta(\boldsymbol{X}_{i,:}), \boldsymbol{Y}_{i,:}), \tag{2.8}$$

where $m << n$ is the size of a mini-batch and $\boldsymbol{X}, \boldsymbol{Y}$ are matrices of samples stacked row-wise. This variant has two advantages: it reduces the noise in the gradient estimate and allows for adjusting the memory complexity of the gradient computation by reducing the size of m. The setting of the learning rate α and other hyperparameters can be determined by out-of-sample performance assessment (e.g., using a validation set to tune them). Repeatedly applying equation 2.8 for an appropriate number of passes over the dataset should produce $\hat{\theta}$ an approximate minimizer of equation 2.4. Given our choice of loss function in equation 2.6 and our assumption about i.i.d training data, $f_{\hat{\theta}}$ can be interpreted as an approximate maximum likelihood estimate of $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x})$. Of course, no guarantees on the goodness of this estimate can be given as the optimization problem is highly non-convex. In practice, however, gradient descent should converge to a reasonably low value of $\mathcal{L}_{bce}$ provided that $\boldsymbol{x}$ contains enough information about the true value of $\boldsymbol{y}$.

2.3 Transformers

While the basic network architecture outlined in the previous section served well for illustration purposes, in practice more complex architectures are used to process sensory information. Today, the Transformer is the deep learning practitioner's architecture of choice. Introduced by Vaswani et al. the transformer is rapidly becoming ubiquitous to almost every application [20, 129, 51, 39, 27, 1, 35]. Originally designed for Natural Language Processing (NLP), the transformer processes sequences of token embeddings (input vectors) in each forward pass. That is, the transformer operates over inputs of the form $\boldsymbol{X} \in \mathbb{R}^{S \times d}$, where S is the set or sequence dimension and d is the feature or channel dimension. The original transformer architecture can be created by stacking N transformer encoder layers, followed by N transformer decoder layers. At the heart of any transformer layer is the attention mechanism it uses. In what follows, we outline two versions of this attention mechanism (sections 2.3.1 and 2.3.2). In section 2.3.3, we discuss details of the transformer encoder block. Finally, sections 2.3.4 and 2.3.5 introduce two adaptations of the transformer architecture used in our empirical study: the vision transformer, and the point transformer.

2.3.1 Dot product attention

Of the many different attention formulations, dot product attention [131] is the most widely used. This mechanism models pair-wise interactions between sequence elements as a fully connected graph, where each layer updates an element's representation based on a convex combination of the elements it is connected to. We note that considering the input to be a sequence, here, implicitly assumes that positional encodings have been added to it to break permutation equivariance. Since the weights of the convex combination are obtained by pushing attention scores through a softmax layer, the most heavily weighted elements are emphasized relative to others—an inductive bias toward only attending to a few elements. Formally, let $\boldsymbol{Q} \in \mathbb{R}^{S' \times d_q}$, $\boldsymbol{K} \in \mathbb{R}^{S \times d_k}$, and $\boldsymbol{V} \in \mathbb{R}^{S \times d_v}$, be the query, key, and value matrices where S and S' are sequence lengths. Then, dot product attention is defined as

$$A(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{2.9}$$

The *softmax* operation is applied row-wise, meaning that each row vector in the output matrix is a convex combination of the rows of $\boldsymbol{V}$. In most settings $d_k = d_v = d_q$. Given an input sequence $\boldsymbol{X} \in \mathbb{R}^{S \times d}$ of length S, the query, key, and value matrices are computed as follows:

$$\boldsymbol{Q} = \boldsymbol{W^Q}\boldsymbol{X}^T,\ \boldsymbol{K} = \boldsymbol{W^K}\boldsymbol{X}^T,\ \boldsymbol{V} = \boldsymbol{W^V}\boldsymbol{X}^T.$$

The operation defined above is known as self-attention (SA) since a single input sequence is used to compute all attention matrices. Note that here $S' = S$. Alternatively, the query matrix may be computed from another sequence $\bar{\boldsymbol{X}} \in \mathbb{R}^{S' \times d}$, allowing related but distinct sequences to attend to each other. This is known as cross-attention (CA) and S need not equal S'. These attention mechanisms can be written using the following shorthand:

$$\mathrm{SA}(\boldsymbol{X}) = A(\boldsymbol{W^Q}\boldsymbol{X}^T, \boldsymbol{W^K}\boldsymbol{X}^T, \boldsymbol{W^V}\boldsymbol{X}^T), \tag{2.10}$$

$$\mathrm{CA}(\bar{\boldsymbol{X}}, \boldsymbol{X}) = A(\boldsymbol{W^Q}\bar{\boldsymbol{X}}^T, \boldsymbol{W^K}\boldsymbol{X}^T, \boldsymbol{W^V}\boldsymbol{X}^T). \tag{2.11}$$

One drawback of dot product attention is its quadratic memory and computational complexity in the sequence length of $\mathcal{O}(S^2)$. This directly results from computing the $S \times S$ attention matrix, which is required to compute the row-wise softmax:

$$A(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_{i,:} = \frac{\sum_{j=1}^{S} \exp(\boldsymbol{Q}_{i,:}\boldsymbol{K}^T_{:,j})\boldsymbol{V}_{:,j}}{\sum_{j=1}^{S} \exp(\boldsymbol{Q}_{i,:}\boldsymbol{K}^T_{:,j})}. \tag{2.12}$$

2.3.2 Linear attention

Linear attention (LA) was proposed in [55] as an alternative to dot product attention, which pivots the computational complexity of the attention operation to be quadratic in the feature dimension $\mathcal{O}(d^2)$ as opposed to the sequence dimension. This is advantageous when dealing with longer sequences or smaller feature dimensions, where $d << S$. This is accomplished by removing the row-wise softmax operation from equation 2.12 in favor of a more efficient way of computing attention weights. Katharopoulos et al. propose using a kernel:

$$\mathrm{LA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})_{i,:} = \frac{\sum_{j=1}^{S} \phi(\boldsymbol{Q}_{i,:})\phi(\boldsymbol{K}_{:,j})^T \boldsymbol{V}_{:,j}}{\sum_{j=1}^{S} \phi(\boldsymbol{Q}_{i,:})\phi(\boldsymbol{K}_{:,j})^T} = \phi(\boldsymbol{Q}_{i,:}) \frac{\sum_{j=1}^{S} \phi(\boldsymbol{K}_{:,j})^T \boldsymbol{V}_{:,j}}{\sum_{j=1}^{S} \phi(\boldsymbol{K}_{:,j})^T}. \tag{2.13}$$

In vectorized form this yields

$$\mathrm{LA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \frac{\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^T \boldsymbol{V}}{\sum_{j=1}^{S} \phi(\boldsymbol{K}_{:,j})^T}. \tag{2.14}$$

From the associativity of matrix multiplication, we see that we can compute the numerator as $\phi(\boldsymbol{Q})(\phi(\boldsymbol{K})^T \boldsymbol{V})$, obtaining the desired complexity. Analogous to dot product attention, we define shorthand for linear self-attention (LSA) and linear cross-attention (LCA),

$$\mathrm{LSA}(\boldsymbol{X}) = \mathrm{LA}(\boldsymbol{W^Q} \boldsymbol{X}^T, \boldsymbol{W^K} \boldsymbol{X}^T, \boldsymbol{W^V} \boldsymbol{X}^T), \tag{2.15}$$
$$\mathrm{LCA}(\bar{\boldsymbol{X}}, \boldsymbol{X}) = \mathrm{LA}(\boldsymbol{W^Q} \bar{\boldsymbol{X}}^T, \boldsymbol{W^K} \boldsymbol{X}^T, \boldsymbol{W^V} \boldsymbol{X}^T). \tag{2.16}$$

2.3.3 Transformer block

We have introduced two variants of the attention operation, which trade-off memory complexity between the sequence length and representation dimension. We will now illustrate how these operations fit into a transformer encoder layer. We leave out the transformer decoder layer for brevity. The output of the lth transformer encoder layer (T_l) can be defined as follows

$$\begin{aligned} I_l =& \mathrm{LN}(\mathrm{MHA}_l(\boldsymbol{X}) + \boldsymbol{X}) \\ T_l(\boldsymbol{X}) =& \mathrm{LN}(h_l(I_l) + I_l) \end{aligned} \tag{2.17}$$

Where LN is layer normalization [2], h_l is an MLP applied to each member of the sequence individually, and multi-head attention MHA is defined as the concatenation of multiple attention operations using different weight matrices but operating over the same input sequence $\boldsymbol{X}$:

$$\mathrm{MHA}_l(\boldsymbol{X}) = \oplus_{i=1}^{h} A_{l,i}(\boldsymbol{W}_{l,i}^{\boldsymbol{Q}} \boldsymbol{X}^T, \boldsymbol{W}_{l,i}^{\boldsymbol{K}} \boldsymbol{X}^T, \boldsymbol{W}_{l,i}^{\boldsymbol{V}} \boldsymbol{X}^T) = \oplus_{i=1}^{h} SA_{l,i}(\boldsymbol{X}), \tag{2.18}$$

where l, i designates the layer l and the attention head i. We note that the dot product self-attention (SA) used here could be replaced with LSA to obtain a transformer with linear attention. The input $\boldsymbol{X}$ to all layers except the first will be the output of the previous layer. The first layer receives $\boldsymbol{X} + \boldsymbol{P}$, the input sequence $\boldsymbol{X}$ added to a positional embedding $\boldsymbol{P}$ that indicated the position of the token in the sequence. Since the self-attention operation is permutation equivariant, $\boldsymbol{P}$ must be added to the sequence. Otherwise, the transformer would become an over-parameterized bag of words model as it would have no way of determining the order of the sequence.

2.3.4 Transformers applied to Image Processing

In their paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", Dosovitskiy et al. propose the vision transformer (ViT) architecture—a transformer architecture for image processing [23]. The authors show their architecture achieves competitive results on image classification compared to SOTA convolutional neural networks(CNNs). To process images with a transformer, the authors split an image into a sequence of evenly sized patches. For instance, an image of resolution $224 \times 224 \times 3$ can be split into 196 patches of size $16 \times 16 \times 3$. These patches are then flattened to $\boldsymbol{X} \in \mathbb{R}^{196 \times 768}$ and added to positional encodings before being processed by a transformer using the exact architecture described above with dot product attention.

2.3.5 Transformers applied to Point-Cloud Processing

Point clouds are unstructured sets living in 3D space. They are, therefore, naturally processed by transformers where elements of the set can be treated as elements of a sequence. Multiple works propose slightly different transformer architectures to process point clouds [51, 167, 1]. However, we will limit our discussion to the adaptation of [51] as it is the only one used throughout our experiments.

The siamese Point-Transformer [51], which we call Point-Transformer (PT) henceforth, computes multi-scale features for each element of the input point-cloud. This is accomplished by as series of self-attention "encoder" blocks that compute features at progressively finer resolutions and subsequent cross-attention "decoder" blocks that upsample the encoder's outputs to compute a representation for each point in the input point cloud.

Encoder For images and text, each element of the input sequence constitutes a meaningful part of the input on its own: a word or an image patch. Although it is natural to treat every point of the point cloud as an element of the input sequence, a single point is only meaningful in relation to the other points in the set. Therefore, it is helpful to aggregate local context within the point cloud to obtain meaningful input features for each point. This can be accomplished using edge convolutions [138] to aggregate information from the k-nearest neighbors of a point in 3D space. Applying edge convolutions to compute features of a set of S points produces a feature map $\boldsymbol{E} \in \mathbb{R}^{S \times d}$ where S, the set dimension, is analogous to the sequence dimension from previous sections and d is the feature or channel dimension.

While PT's architecture is similar to the original transformer it is not identical like ViT. For instance, PT does not stack encoder layers by directly taking the output of the previous layer. Instead, PT's encoder layers operate over edge features $\boldsymbol{E}_l$ of progressively coarser resolution. The edge features input to each layer are computed as described above, by computing features of KNN neighbourhoods for points processed by the previous encoder layer. Each of these subsamples contains $\frac{N}{2^l}$ points, where N is the number of input points. In this way, features are passed from one encoder layer to the next when they are aggregated within the edge convolution, leading to progressively coarser features as each encoder summarizes the features of the previous layer using fewer centroids.

During the forward pass, PT first encoder takes as input the set of point features $\boldsymbol{E}_l$ and corresponding positional embeddings $\boldsymbol{P}_l$ that are computed directly from the 3D points. The PT's encoder layer uses multi-head linear self-attention defined as follows:

$$\mathrm{MHA}_l^{E_{pts}}(\hat{\boldsymbol{E}}_l) = \oplus_{i=1}^{h} \mathrm{LSA}_{l,i}(\hat{\boldsymbol{E}}_l), \tag{2.19}$$

where $\hat{\boldsymbol{E}}_l = \boldsymbol{E}_l + \boldsymbol{P}_l$. Then, PT's transformer encoder layer can then be defined as

$$\begin{aligned} I_l =& \mathrm{LN}(h_l(\mathrm{MHA}_l^{E_{pts}}(\hat{\boldsymbol{E}}_l))), \\ T_l^{pts}(\hat{\boldsymbol{E}}_l) =& \mathrm{LN}(\mathrm{MLP}_l(I_l \oplus \boldsymbol{E}_l)) + I_l, \end{aligned} \tag{2.20}$$

where h_l is a linear layer, MLP_l is a multilayer perceptron, LN is a layer nomalization, and $\oplus$ designates feature level concatenation. After applying these encoder layers to different resolution input features we obtain features maps $\boldsymbol{H}_l = T_l^{pts}(\hat{\boldsymbol{E}}_l)$ for $l \in \{1, 2, 3\}$.

Decoder To propagate the encoded information from features at coarser resolutions to finer resolutions, the authors use a version of multi-head cross-attention between subsequent encoded feature maps computed as follows:

$$\mathrm{MHA}_l^{D_{pts}}(\boldsymbol{H}_l, \boldsymbol{O}_{l+1}) = \oplus_{i=1}^{h} \mathrm{LA}(\boldsymbol{W}_{l,i}^{\boldsymbol{Q}} \boldsymbol{H}_l^T, \boldsymbol{W}_{l,i}^{\boldsymbol{K}} \boldsymbol{O}_{l+1}^T, \boldsymbol{W}_{l,i}^{\boldsymbol{V}} (\boldsymbol{O}_{l+1} + \boldsymbol{P}_{l+1})^T). \tag{2.21}$$

The corresponding decoder block is defined similarly to the encoder:

$$\begin{aligned} I_l =& \mathrm{LN}(h_l(\mathrm{MHA}_l^{D_{pts}}(\boldsymbol{H}_l, \boldsymbol{O}_{l+1}))), \\ \boldsymbol{O}_l =& \mathrm{LN}(\mathrm{MLP}_l(I_l \oplus \boldsymbol{H}_l)). \end{aligned} \tag{2.22}$$

This is done for $l = 2, 1, 0$, where $\boldsymbol{O}_3 = \boldsymbol{H}_3$ and $\boldsymbol{H}_0$ is the original set of points in stacked matrix form. Therefore, the final output $\boldsymbol{O}_0$ is a feature map for each point in the set, where each feature depends on the coarser-resolution features of previous layers.

Chapter 3

Related Work

This section briefly outlines areas relevant to our study of object ReID from point clouds and highlights the absence of point-based ReID in 3D multi-object tracking, thus far.

3.0.1 Point-Processing Networks

The ability to effectively represent irregular sets of points is essential for 3D geometry processing. Respecting the symmetries of permutation invariance (PointNet) [101] and the metric space structure of raw point clouds (PointNet++) [102] were shown early on to be important priors. Subsequent works propose edge convolutions to process point clouds in CNN-style [138], a performant and efficient residual-MLP framework [81], exploiting the benefits of depth [62], using the MLP-Mixer [16], using a transformer-based architecture [167], among others. In the following study, we select three efficient models to use for our experiments: PointNet [101], DGCNN [138], and Point-Transformer [51].

3.0.2 3D Single Object Tracking

Single object tracking (SOT) from point clouds focuses on the task of identifying a single target object within a large search area. Consequently, most methods apply point processing networks to compare the target to the search area, techniques that could be adapted to our point-cloud-ReID-for-MOT setting. Indeed, many recent works [32, 51, 50, 117, 103] have shown the benefits of siamese point-processing networks for SOT. Giancola et al. [32] learn a similarity function between cropped point cloud patches and use a Kalman filter to generate candidate bounding boxes for matching with the current target observation. In follow-up work, P2B [103] eliminates the need to approximate greedy search at inference time in favor of an end-to-end regression approach that directly estimates the target's next position through Hough voting. Hui et al. [50] improve on this approach with a novel regression head inspired by work on object detection [29]. Specifically, they use point features computed from a siamese network which learns target and search area features together as input to a voxelization branch, which voxelizes the 3D space and pools along the height dimension. A 3D object-detection-style head is then applied over the resulting BEV feature map to regress center position, height, and yaw. In their latest work [51], the authors show that their results can be improved by employing a Point-Transformer backbone. Given the strong performance of their architecture for SOT, we include it in our study and show that the Point-Transformer also performs strongly for re-identification in a multi-object tracking context.

3.0.3 3D Multi-Object Tracking

Maintaining 3D tracklets for detected objects is an essential task for autonomous vehicles—allowing them to forecast the future motion of these objects for planning. Many 3D MOT approaches follow the tracking-by-detection paradigm [140, 15, 160, 67, 163, 40, 57, 110, 142, 66, 98, 133, 141, 58], where an object detector is used to identify candidate objects for matching to existing tracks and a motion model is used to propagate tracks from one frame to the next. Detections from LiDAR-based object detectors are of high quality, simplifying the downstream tracking task as motion cues are more reliable with better detections. However, approaches that additionally use appearance-based features [142, 58, 163, 133], typically outperform other methods for a given object detector. Of all the works mentioned, only one [141] leverages representations of object-level point clouds cropped from their predicted 3D bounding boxes. While we also crop object point clouds from their 3D bounding boxes, our work differs in multiple ways: 1) the other work evaluates the proposed networks with respect to the 3D MOT task, while our results focus on point cloud ReID accuracy compared to image ReID, 2) the results in [141] are only for the KITTI-car split, while we evaluate our model's performance on all classes for nuScenes and WOD, and 3) the other work does not provide details on how the points used for ReID are extracted. Sufficient details are needed to address our main research question, however. If the points are not normalized, then a point processing network could learn a motion model.

3.0.4 Object Re-Identification from images and point clouds.

ReID from images has been an active area of research for many years, with most work focusing on Vehicle ReID [175, 170, 114, 25, 28, 73, 56] or Person ReID [169, 18, 48, 52, 53]. In contrast, ReID which leverages depth sensor information has received relatively little attention. A number of works consider ReID from RGB-D data [87, 68, 72, 99], leveraging the depth information in conjunction with skeleton normalization to improve re-identification performance. A recent work [171] proposes OG-NET for pedestrian ReID from point clouds. However, the work uses point clouds that are artificially generated from image datasets for person re-identification, using a human pose estimation pipeline applied to each image. In contrast, our study involves real observations cropped directly from large-scale autonomous driving datasets.
Finally, in what is the closest work to our own, Giancola et al. [32] propose an improvement to 3D single-object tracking (SOT) which leverages shape completion regularization for similarity learning. Specifically, they use a siamese network and train it to encode cropped (e.g., using GT bounding boxes) point cloud observations into a latent vector representation similar to one of the corresponding complete object (created by aggregating points from all the crops). At inference, they use a Kalman filter to generate candidate bounding boxes for matching to the current target observation. While this method also proposed to train ReID networks from cropped bounding box observations, our studies differ in multiple ways: 1) they train their networks for the 3D SOT task, while our focus is 3D MOT, 2) their study only involves cars and is limited to the KITTI dataset, while we train and evaluate on all classes of the large-scale nuScenes and Waymo datasets, and 3) their study does not compare performance with image re-identification models.

Chapter 4

ReID Dataset Construction

To train our point re-identification networks, we extract object observations from the nuScenes dataset [7] and the Waymo Open Dataset (WOD) [126]. This extraction process seeks to maximize the applicability of our results to a tracking-by-detection context and is non-trivial. Here, we briefly describe the salient details. Starting with an overview of each dataset's sensor suite, we highlight the differences between them and explain how we mitigated their class discrepancy. We then detail the object detectors used to obtain ReID samples with realistic noise and explain our object extraction procedure for point clouds and images. Finally, we discuss the statistics of our newly created ReID datasets and outline our training-time and testing-time sampling algorithms.

4.1 Dataset differences

Sensors Each dataset contains multimodal driving data captured from one (nuScenes) or multiple (WOD) LiDAR sensors and an array of camera sensors. nuScenes employs a single 360° 32-beam LiDAR, while WOD features four close proximity LiDAR sensors on its front, back, and sides and one 64-beam 360° 10 Hz top-mounted sensor. This means that LiDAR scans from the WOD will have a much better coverage of the entire scene than their nuScenes counterparts (see fig.4.1), allowing us to study how point ReID improves as sensor resolution increases. The situation is reversed for cameras, however. On nuScenes, six cameras capture a full 360° view of the scene, while the WOD was captured from a vehicle equipped with five cameras that have a front-facing field of view (FOV) of $\sim 252°$ and a corresponding blind spot behind the vehicle. The nuScenes cameras capture six 1600×1200 images, while the WOD cameras are of slightly higher resolution, producing

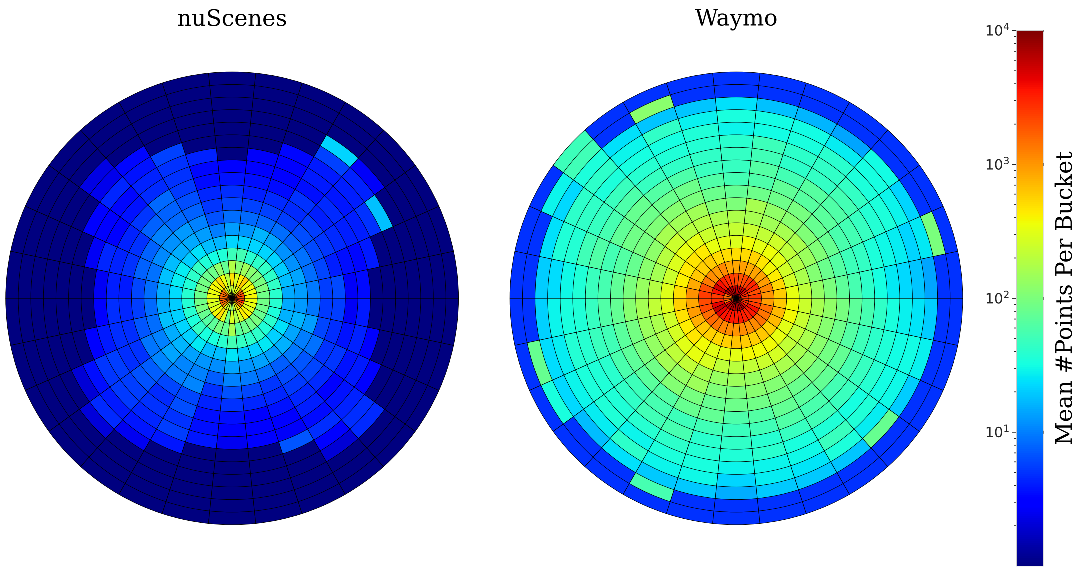

Figure 4.1: **WOD observations (64-beam LiDAR) are much denser than their nuScenes counterparts (32-beam LiDAR).** The figure shows a heatmap of the mean number of points per bucket for the nuScenes (left) and Waymo (right) datasets. The mean is calculated from the number of points within true positive bounding boxes whose center lies within the bucket. The 30 distinct azimuth ranges shown cover 360° and the 18 radii shown are each five meters apart. The heatmaps are centered on the ego vehicle.

three 1920×1280 frontal images and two 1920×886 lateral images. The WOD vehicle's LiDAR-Camera FOV discrepancy is a perfect illustration of a real-world robotics system that can benefit from the added complementarity of LiDAR-based ReID.

Class Labels The nuScenes dataset annotates 10 distinct classes (see table 4.2); however, we only concern ourselves with the classes considered on their tracking benchmark: car, bus, pedestrian, truck, bicycle, motorcycle, and trailer. In their original dataset release, WOD provided substantially fewer tracking labels: vehicle, pedestrian, and bicycle. In an effort to increase the compatibility of both datasets, we enhance the original WOD class labels using their point cloud segmentation labels (released in a subsequent update to the dataset). Specifically, we annotate the objects within segmentation-annotated frames using the majority vote of annotated points within a bounding box. Then, using the tracking labels, we propagate the new class of the object to all frames. This procedure expands the labels to include car, truck, bus, and motorcycle. We note that some objects don't receive a new label as they never occur in a segmentation-annotated frame. We choose to ignore these objects as they make up a negligible amount (0.6%) of all annotated objects (see the "other_vehicle" class in figure 4.2).

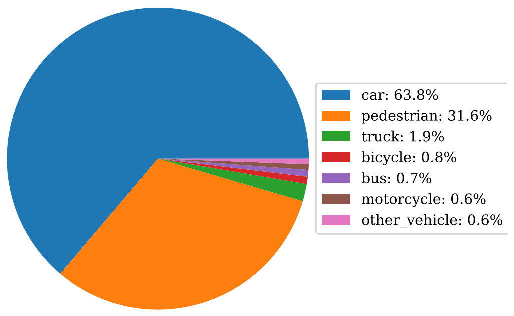

Figure 4.2: **Waymo class label split after propagating segmentation annotations.** The plot shows the proportions of different types of objects in the WOD training set after propagating segmentation labels. We note that the samples belonging to the "other_vehicle" class are ignored during the creation of our ReID datasets, since we do not have label information for them.

4.2 Object extraction

Obtaining bounding boxes from 3D object detectors To simulate the noise encountered in a real tracking-by-detection setting we extract object observations using predicted bounding boxes from 3D object detectors. This improves noise realism over previous work that simply applied Gaussian noise to GT bounding boxes' position and yaw [32]. To extract these realistic observations, we use ground-truth tracking annotations in conjunction with bounding boxes predicted from 3D object detectors. For the nuScenes dataset, we use detections from the pre-trained BEVfusion C+L checkpoint provided by [75]. For WOD, we train our own object detector using the CenterPoint [155] implementation provided by MMDetection3D [19]. Our CenterPoint model is trained for 20 epochs, where each epoch constitutes a pass over 20% of the frames in WOD (sampled uniformly at random). We report the performance of each detector on their respective validation sets in tables 4.1 and 4.2. While results between datasets are difficult to compare, we hypothesize that the nuScenes model attains relatively stronger performance. This is to be expected as the model incorporates both camera and LiDAR information, while our Waymo model does not. We note that this performance difference should have an impact on the level of noise in predicted bounding boxes and consequently may impact our ReID network's performance.

Class	**Level 1**		**Level 2**	
	mAP	mAPH	mAP	mAPH
Car	68.3	67.8	60.7	60.2
Truck	34.2	33.5	32.2	31.6
Bus	49.0	48.6	42.6	42.3
Motorcycle	53.1	52.2	38.0	37.4
Cyclist	68.4	67.0	65.8	64.5
Pedestrian	65.9	59.7	58.0	52.4

Table 4.1: **Performance of the CenterPoint model on the WOD with enhanced vehicle labels.** The metrics are only calculated on samples with an enhanced label.

Class	**AP**	**ATE**	**ASE**	**AOE**	**AVE**	**AAE**
Car	89.2	0.170	0.148	0.061	0.274	0.185
Truck	64.6	0.326	0.181	0.093	0.247	0.217
Bus	75.3	0.338	0.189	0.069	0.430	0.274
Trailer	42.5	0.520	0.201	0.610	0.214	0.140
Const. Veh.	30.4	0.735	0.431	0.797	0.118	0.295
Pedestrian	88.2	0.134	0.288	0.387	0.217	0.101
Motorcycle	78.6	0.184	0.249	0.216	0.348	0.271
Bicycle	65.1	0.169	0.257	0.411	0.190	0.015
Traffic Cone	79.5	0.121	0.317	-	-	-
Barrier	72.0	0.178	0.277	0.054	-	-
NDS: 0.714	68.5	0.288	0.254	0.300	0.255	0.187

Table 4.2: **Performance of the BEVfusion C+L model on the nuScenes validation set.** This model was used to generate the nuScenes detections for tracking and to create the nuScenes ReID dataset.

Extracting objects Each detector implements non-maximal suppression before outputting its detections—an important step in most tracking-by-detection pipelines. Our object extraction pipeline further eliminates noisy detections by thresholding their confidence score to be above $\tau_c = 0.1$. Using the remaining detections, we extract true and false positives by matching detected bounding boxes to ground truth bounding boxes using a permissive intersection over union (IoU) threshold of $\tau_{IoU} = 0.01$. That is, true positives correspond to predicted bounding boxes of the correct class whose IoU with the corresponding ground truth bounding box is above τ_{IoU}. Hungarian matching is used here to obtain a unique assignment between ground truth and true positive bounding boxes. Duplicate true positives are discarded. Conversely, false positives are obtained by extracting observations whose IoU with the ground truth is below τ_{IoU}. To extract observations from LiDAR scans, we first crop points within an object's 3D bounding box before translating and rotating them such that the 3D bounding box becomes centered at $(0, 0, 0)^T$ and faces a canonical orientation. Note that despite this normalization step, the observations will still contain noise from the object detector's prediction; that is, the object's true orientation will not necessarily be facing the canonical orientation, nor will its true center necessarily be at $(0, 0, 0)^T$. To extract observations from images, we project the predicted 3D bounding boxes to the image plane using the camera's projection matrix. Depending on the relative angle of the 3D bounding box to the image plane, the bounding box may project to a non-rectangular shape on the image plane. To obtain a rectangular 2D bounding box, we always use the bounding box enclosing the projected shape. For bounding boxes that project to multiple images, we select the projection with the largest enclosing bounding box.

Complete objects In addition to collecting sparse point cloud observations, we also obtain a dense point cloud for each object. Similar to previous work [83, 134, 32], we use each dataset's tracking annotations to aggregate points for an object within all of its ground truth bounding boxes. Each individual point cloud observation is rotated and translated such that its bounding box is centered at the origin and facing a canonical orientation—this time, there will be little noise since we use ground truth bounding boxes. To further enhance our dense point cloud, we exploit object's symmetry and mirror the point cloud across the object's centerline. Finally, all points are concatenated to form one dense point cloud.

4.3 Dataset statistics and sampling algorithms

Statistics Table 4.3 reports the training split statistics for our ReID datasets created from nuScenes (left) and WOD (right). Our dataset extraction pipeline collects $35,212$ unique true positive objects on nuScenes and $63,756$ on WOD. Each unique object can have one or more observations. On nuScenes, individual observations total $856,013$, while WOD has many more observations totaling $6,899,297$. To train our networks for the re-identification task, we sample positive and negative pairs of observations. When sampling these pairs, however, we only sample objects from the same class. We chose to do this for two reasons: 1) during training, our networks focus on difficult comparisons—objects of different classes are often easy to distinguish—and 2) this enables reduced inference overhead in a tracking-by-detection context, by only comparing objects of the same predicted class. Even without combining objects of different classes, the number of possible positive and negative pairs is enormous. On nuScenes, there are over six million positive pairs, while WOD contains 435 million. However, this is nothing compared to the number of negative pairs: $3.9e + 16$ and $3.98e + 19$ for nuScenes and WOD respectively. To put this scale in perspective, the WOD ReID dataset has 11 orders of magnitude more samples than the popular ImageNet[22] dataset. We will, therefore, be training networks in the underfitting regime; that is, we will be unable to overfit to the training set. We also note that both ReID datasets feature a severe imbalance problem. For instance, on WOD the positive pairs make up just $1.12e-9\%$ of the total number of possible pairs.

Classes	# of Obj.	# of Obs.	Pos. Pairs	Neg. Pairs
FP bicycle	$--$	13,100	$--$	$--$
FP bus	$--$	2,069	$--$	$--$
FP car	$--$	125,223	$--$	$--$
FP motorcycle	$--$	7,222	$--$	$--$
FP pedestrian	$--$	106,206	$--$	$--$
FP trailer	$--$	8,023	$--$	$--$
FP truck	$--$	21,792	$--$	$--$
bicycle	554	7,575	$7.76e+4$	$1.62e+12$
bus	422	8,375	$1.07e+5$	$4.54e+11$
car	20,830	326,967	$3.77e+6$	$3.34e+16$
motorcycle	588	8,201	$8.67e+4$	$9.73e+11$
pedestrian	9,112	156,852	$1.83e+6$	$5.43e+15$
trailer	773	13,921	$1.66e+5$	$3.34e+12$
truck	2,933	50,487	$5.98e+5$	$1.32e+14$
Total	35,212	856,013	$6.63e+06$	$3.90e+16$

Classes	# of Obj.	# of Obs.	Pos. Pairs	Neg. Pairs
FP bicycle	$--$	27,043	$--$	$--$
FP bus	$--$	2,378	$--$	$--$
FP car	$--$	24,971	$--$	$--$
FP motorcycle	$--$	46,111	$--$	$--$
FP pedestrian	$--$	604,021	$--$	$--$
FP truck	$--$	5,987	$--$	$--$
bicycle	497	46,204	$3.02e+6$	$1.23e+14$
bus	378	43,639	$3.35e+6$	$4.59e+13$
car	43,305	4,002,934	$2.78e+8$	$3.25e+19$
motorcycle	355	32,317	$2.18e+6$	$9.90e+13$
pedestrian	18,107	1,954,970	$1.41e+8$	$6.40e+18$
truck	1,114	108,722	$7.77e+6$	$7.13e+14$
Total	63,756	6,899,297	$4.35e+08$	$3.89e+19$

Table 4.3: **Training set statistics for nuScenes (left) and WOD (right).** From left to right, the columns contain the number of unique objects of each class, the number of observations of these objects, the number of positive pairs of each class, and the number of negative pairs of each class. False positives are used to create negative pairs with observations of the corresponding true positive class. Positive and negative pairs are created from objects of the same predicted class.

Sampling at training time Our training-time sampling procedure is detailed in algorithm 1. At training time, one epoch constitutes one pass over every unique object in the dataset. For each object O (of class c), we flip a coin to determine whether to sample an associated positive or negative observation. Positives are sampled uniformly at random from the other observations of O, while choosing to sample negative leads to another coin toss. This time, we select between sampling a false positive FP of class c' (c' denotes a false positive misclassified by the object detector as belonging to class c) or a true positive by sampling an object O' of class c other than O and then sampling from its observations uniformly at random. Therefore, during training, our networks learn to predict a match or no match between objects of the same predicted class.

Sampling at testing time At testing time, we sample a balanced test set of reasonable size that can reveal the true performance of our models at all point densities. To accomplish this, we sample at most 10 distinct positive pairs (o_1,o_2) for each object in the test set,

Eval set	Positive Pairs	Negative Pairs	Total
nuScenes eval	53,834	53,686	107,520
waymo eval 1	119,874	119,742	239,616
waymo eval 2	145,470	145,346	290,816

Table 4.4: **Evaluation set statistics.**

keeping track of their point densities(d_{o_1},d_{o_2}). Then, for each positive pair (o_1,o_2), we sample a corresponding negative pair (o_1,o'_2), where o'_2 has a similar point density to o_2. We define point densities as similar if they fall within the same power-two interval: $[2^n, 2^{n+1})$. Before sampling from the nuScenes test set, we filter out observations that have fewer than two points and observations without image crops. We name this test set *nuScenes eval*. On WOD, we create two test sets. The first, called *Waymo eval 1*, is created identically to *nuScenes eval*. The second, called *Waymo eval 2*, only filters out observations that have fewer than two points. Therefore, it will include many observations that have no associated image crops as they are out of the sensor's field of view, exposing the actual performance of the image models. These test sets are each sampled with the seed set to 66 so that the same samples are used every time. Table 4.4 reports the statistics for these evaluation sets.

Algorithm 1: Data Sampling Algorithm for one Epoch of Training

```
Data: Dataset $D$ of all objects and their observations.
Result: A set of sampled pairs for one epoch of training.
// Initialize sample
$S \leftarrow \emptyset$;
foreach object $O$ in $D$ do
    $o_1 \leftarrow$ random observation of $O$;
    $c \leftarrow$ class of $O$;
    $p_1 \leftarrow$ random number between 0 and 1;
    if $p_1 \leq 0.5$ then
        // Sample positive pair
        $o_2 \leftarrow$ random observation of $O$ other than $o_1$;
        $l \leftarrow 1$;
    end
    else
        // Sample negative observation
        $l \leftarrow 0$;
        $p_2 \leftarrow$ random number between 0 and 1;
        if $p_2 \leq 0.5$ then
            // Sample false positive
            $o_2 \leftarrow$ random false positive of predicted class $c$;
        end
        else
            // Sample true positive
            $O' \leftarrow$ random object of class $c$ other than $O$;
            $o_2 \leftarrow$ random observation of $O'$;
        end
    end
    $S \leftarrow S \cup (o_1, o_2, l)$;
end
return $S$
```

Chapter 5

Method and Experiments

This chapter is divided into three parts. We first introduce our proposed matching head for real-time ReID and provide the objective we use to train it. Next, we introduce the architectures used throughout our empirical studies and report their training details. Finally, we report and discuss the results of four empirical studies: a comparison of point and image ReID on nuScenes and WOD, an investigation of the effects of scaling training time on WOD, an investigation of our ReID network's ability to complement strong motion-based trackers on the nuScenes dataset, and an ablation study that applies popular representation learning techniques to our problem.

5.1 Method

In the following section, we present our proposed real-time matching module (RTMM) for making efficient pairwise comparisons in a 3D multi-object-tracking-by-detection context. We then illustrate how existing point-based architectures can be adapted to use it—leading to a family of RTMM-based point cloud ReID networks and we define our training objective.

5.1.1 A real-time matching mechanism for point sets

Given our goal of evaluating point cloud ReID in a multi-object-tracking-by-detection context, it is important for our matching module to be capable of making a large number of pairwise comparisons in real-time (e.g., between tracks and detections from one timestep to the next). Inspired by the strong performance of Hui et al (2022)'s Point-Transformer for

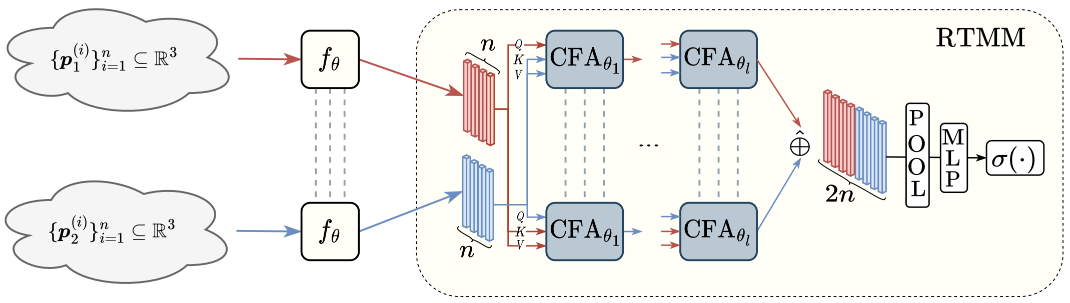
$\{\boldsymbol{p}_1^{(i)}\}_{i=1}^n \subseteq \mathbb{R}^3$
$\{\boldsymbol{p}_2^{(i)}\}_{i=1}^n \subseteq \mathbb{R}^3$
f_θ
f_θ
RTMM
n
n
Q
K
V
CFA_{θ_1}
CFA_{θ_l}
...
$\hat{\oplus}$
$2n$
POOL
MLP
$\sigma(\cdot)$

Figure 5.1: **The architecture of our proposed matching head, RTMM.** RTMM compares two vector-valued input sequences, making it compatible with various sequence or set processing backbones.

single object tracking, we adapt their "coarse-to-fine correlation network" to fit our needs. Specifically, we eliminate the ego feature augmentation module, as we found its memory complexity scales poorly with the number of comparisons being made, making a large number of real-time comparisons infeasible. As a result, we only keep their cross-feature augmentation (CFA) module, which uses efficient linear attention [55]. While we maintain the same structure within CFA blocks as [51] (described below), we modify the structure of their correlation network to make it symmetric, exploiting the inherent symmetry of our pairwise matching problem. To make symmetric comparisons between two sets of points $\{\boldsymbol{x}_1^{(i)}\}_i^{n_1}, \{\boldsymbol{x}_2^{(i)}\}_i^{n_2}$, with $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$, we apply $l = 2$ CFA blocks between each point cloud's representation and concatenate the resulting features maps along the sequence dimension, before applying an invariant pooling operation along it (see fig. 5.1). Finally, the outputs are fed to a residual MLP for binary classification of match or no match. Concretely, CFA blocks receive two sets of points $\{\boldsymbol{x}_1^{(i)}\}_i^{n_1}, \{\boldsymbol{x}_2^{(i)}\}_i^{n_2}$, which we designate in stacked matrix form $\boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbb{R}^{n\times 3}$ henceforth. The point sets are subsampled or resampled to size n (we use $n = 128$ in all our experiments) before being fed to a point backbone f_θ to compute representations $f_\theta(\boldsymbol{X}_1), f_\theta(\boldsymbol{X}_2)$. We note that f_θ can be any set or sequence processing network. A CFA block first computes linear cross-attention (LCA):

$$\boldsymbol{L} = \mathrm{LCA}(f_\theta(\boldsymbol{X}_1), f_\theta(\boldsymbol{X}_2) + \boldsymbol{P}) \tag{5.1}$$

where $\boldsymbol{P} = \mathrm{MLP}_{pos}(\boldsymbol{X}_2)$ is a positional encoding computed from the corresponding 3D points. Next, a layer normalization (LN) is applied followed by an MLP applied to the channel-wise concatenation of $\mathrm{LN}(\boldsymbol{L})$ and $f_\theta(\boldsymbol{X}_1)$ and another layer normalization is applied,

$$\boldsymbol{L}' = \mathrm{LN}(\ \mathrm{MLP}(\ \mathrm{LN}(\boldsymbol{L}) \oplus f_\theta(\boldsymbol{X}_1))). \tag{5.2}$$

Model	Parameters	Batch Size	Inference Time
RTMM	223K	2000	49.7 ms $\pm$ 5.84 ms
Point Baseline	132K	2000	0.19 ms $\pm$ 0.01 ms
DeiT-Tiny	5910k	100	32.4 ms $\pm$ 1.75ms
DeiT-Base	87,338k	100	173 ms $\pm$ 11.8ms
DGCNN	617k	100	18.7 ms $\pm$ 3.25ms
PointNet	2800k	100	8.04 ms $\pm$ 0.099ms
Point-Transformer	529k	100	15 ms $\pm$ 0.605ms
Point-Transformer2M	2000k	100	27.5 ms $\pm$ 0.149ms
Point-Transformer4M	4413k	100	43.4 ms $\pm$ 0.435ms
Point-Transformer7M	7768k	100	58 ms $\pm$ 0.796ms

Table 5.1: **Inference speed of different point and image backbones used in our experiments.** All models were tested on a single RTX 3090 GPU.

Finally, a residual connection is applied to complete the CFA block:

$$\mathrm{CFA}(f_\theta(\boldsymbol{X}_1), \boldsymbol{X}_1, f_\theta(\boldsymbol{X}_2), \boldsymbol{X}_2) = \boldsymbol{L}' + f_\theta(\boldsymbol{X}_1). \tag{5.3}$$

Then, our real-time matching module is structured as follows:

$$\bar{\boldsymbol{X}}_1^l = \mathrm{CFA}_{\theta_l}(\bar{\boldsymbol{X}}_1^{l-1}, \boldsymbol{X}_1, \bar{\boldsymbol{X}}_2^{l-1}, \boldsymbol{X}_2) \tag{5.4}$$

$$\bar{\boldsymbol{X}}_2^l = \mathrm{CFA}_{\theta_l}(\bar{\boldsymbol{X}}_2^{l-1}, \boldsymbol{X}_2, \bar{\boldsymbol{X}}_1^{l-1}, \boldsymbol{X}_1) \tag{5.5}$$

$$\mathrm{RTMM}_\theta^l(\boldsymbol{X}_1, \boldsymbol{X}_2) = \sigma(\mathrm{MLP}_{res}(\mathrm{pool}(\bar{\boldsymbol{X}}_1^l \hat{\oplus} \bar{\boldsymbol{X}}_2^l))), \tag{5.6}$$

where MLP_{res} is a residual MLP block followed by a linear projection layer (mapping each output to $\mathbb{R}$); $\mathrm{pool}(\boldsymbol{x}) := \mathrm{maxpool}(\boldsymbol{x}) \oplus \mathrm{avgpool}(\boldsymbol{x})$; $\hat{\oplus}$ designates sequence/set level concatenation; $\oplus$ designates vector concatenation of the channel dimension; $\bar{\boldsymbol{X}}_i^0 = f_\theta(\boldsymbol{X}_i)$; and $\sigma(\cdot)$ is the sigmoid activation function. In practice, we find that setting $l = 2$ is sufficient to achieve strong matching performance.

We note that on all datasets and for all point models, we subsample or resample the input point cloud $\boldsymbol{X}$ to contain 128 points. This decision was made to avoid unnecessarily increasing the sequence length (and thus the inference time) when most observations from nuScenes and Waymo have fewer than 128 points. While some may argue that this would limit the performance of our networks at higher point densities, our experiments show (see figure 5.3) that our networks' performance continues to increase for objects at point densities well above 128.

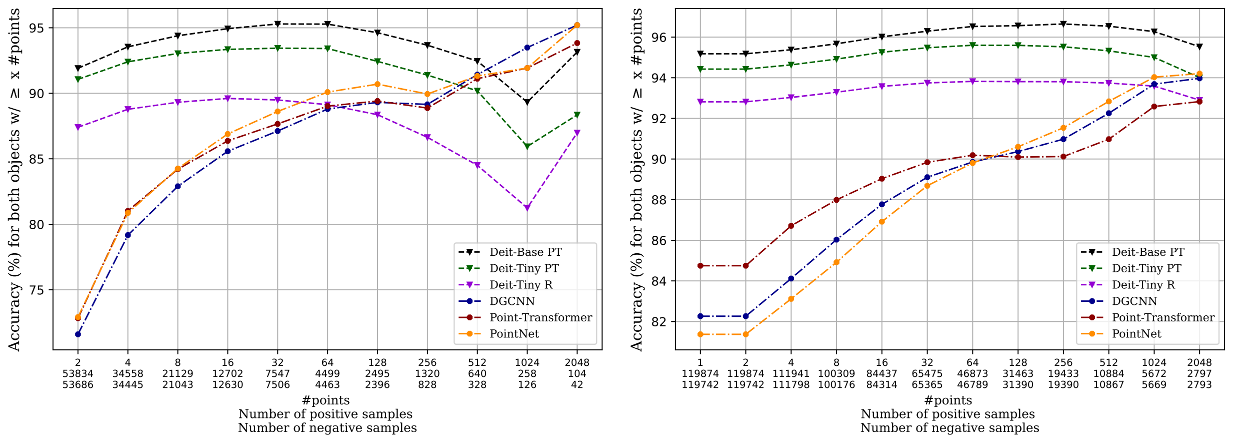

Figure 5.2: **The performance of point cloud ReID approaches image ReID with sufficient points.** The plot show the performance of the image and point cloud ReID networks as a function of the number of points in the observations. Left is the *nuScenes eval* set, while right is *Waymo eval 1*.

5.1.2 Compatibility with existing point backbones

In our empirical evaluation, we select f_θ to be one of the following: PointNet[101], DGCNN[138], and Point Transformer [51]. However, almost any point processing backbone can be adapted with minimal effort to use our proposed RTMM. Due to the unstructured nature of point cloud inputs, most point-processing backbones compute an intermediate representation $f_\theta(\boldsymbol{X}) \in \mathbb{R}^{N\times d}$ that is equivariant to permutations of the columns of $\boldsymbol{X}$, followed by an invariant pooling layer. Such constructions preserve the set cardinality dimension N until the pooling operation, making them amenable to processing using sequence models such as our RTMM. Therefore, many existing point backbones can be adapted to our method by extracting their representation before invariant pooling layers.

5.1.3 Training objective

We train our networks for object re-identification tasks using binary cross-entropy (5.7) but experiment with various other objectives from the literature.

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i \cdot \log(\boldsymbol{x}_i) + (1-\boldsymbol{y}_i)\log(1-\boldsymbol{x}_i)) \tag{5.7}$$

We find, however, that binary cross-entropy achieves results stronger or on par with other formulations while being simple and fast to train with. We discuss further details of other formulations in our ablation study (Table. 5.4 and sec. 5.6).

5.2 Experimental Setup

Models Compared To place our experiments within a meaningful context, we train three image models and one point cloud baseline model to compare with our three point cloud ReID networks (PointNet [101], DGCNN [138], and Point-Transformer [51]). For our image baselines, we select DeiT [129], a family of efficient vision transformers of different sizes. They are efficient and can be adapted with little effort to use our proposed RTMM, unlike CNNs. Specifically, we choose DeiT-Tiny as our main point of comparison and train one DeiT-Tiny model from a pre-trained checkpoint and another from random initialization. DeiT-Tiny allows us to assess the performance of an image model with a *comparable* number of parameters to our point models (5M v.s. 2.8M). We also train a larger DeiT-Base model from a pre-trained checkpoint for reference. To compare RTMM to an alternative point-processing model, we train a naive point baseline. It uses a Point-Transformer backbone to compute representations of the compared point clouds and concatenates the pooled representations, feeding them to a residual MLP for matching: $\mathrm{MLP}'_{res}(\mathrm{pool}(f_\theta(\boldsymbol{X}_1)) \oplus \mathrm{pool}(f_\theta(\boldsymbol{X}_2)))$.

Inference Speed Table 5.1 compares the inference speed of the different architectures in our study. The first two entries correspond to different matching heads: the baseline and our proposed RTMM. We see that the baseline is two orders of magnitude faster than our proposed RTMM; however, our experiments show (see table 5.2) the performance difference is well worth the added latency. The remaining entries compare different backbone architectures which can precede either of the matching heads. We note that we compare match heads and backbones with different batch sizes of 2000 and 100, respectively. This is because the backbones process samples individually, while the matching heads process pairs of samples. Thus, for multi-object tracking the backbone need only process all detections at the current timestep (which is often below 100), while the matching head must make pairwise comparisons between two timesteps, significantly increasing the batch size. The backbone models are divided into three sections: image models, the models used in our image versus point cloud experiments (table 5.2), tracking error study (figure 5.6), and ablation study 5.7, and models used in our scaling experiments. DeiT-Tiny, the smallest

Backbone	Par.	Acc.	F1 Pos.	F1 Neg.	Car	Pedestrian	Bicycle	Bus	Motorcycle	Truck	FP
DeiT-base^{*I}	85.7M	**91.91**%	**91.75**%	**92.06**%	**94.24**%	**88.17**%	**89.97**%	**92.53**%	**89.14**%	**91.28**%	**95.67**%
DeiT-tiny^{*I}	5.7M	91.06%	91.07%	91.06%	93.43%	87.37%	88.97%	91.35%	88.44%	90.31%	93.36%
DeiT-tiny^{I}	5.7M	87.41%	87.52%	87.29%	90%	83.2%	84.24%	86.18%	84.57%	86.98%	89.43%
DGCNN^{L}	0.6M	71.6%	70.75%	72.41%	75.47%	61.68%	63.66%	77.88%	65.08%	79.11%	80.09%
Point-Transformer^{L}	0.5M	72.83%	72.1%	73.52%	77.21%	61.89%	64.37%	79.5%	65.45%	80.56%	80.51%
Pointnet^{L}	2.8M	72.92%	72.19%	73.6%	76.85%	63.12%	64.09%	78.12%	65.69%	80.27%	80.77%
Point-Baseline^{L}	0.5M	60.8%	59.94%	61.61%	64.28%	52.55%	51.81%	66.52%	52.31%	65.53%	65.12%
DeiT-base^{*I}	85.7M	**95.18**%	**95.17**%	**95.19**%	**96.01**%	**93.59**%	**94.03**%	**90.33**%	90.02%	**93.56**%	**97.27**%
DeiT-tiny^{*I}	5.7M	94.43%	94.47%	94.38%	95.26%	92.77%	93.59%	89.54%	**91.5**%	92.96%	95.87%
DeiT-tiny^{I}	5.7M	92.82%	92.92%	92.71%	93.87%	90.71%	91.97%	86.79%	89.4%	91.07%	93.14%
DGCNN^{L}	0.6M	82.26%	82.09%	82.43%	84.65%	76.89%	80.24%	79.8%	68.67%	87.91%	88.08%
Point-Transformer^{L}	0.5M	84.75%	84.76%	84.73%	86.96%	79.9%	79.04%	85.85%	73.66%	87.71%	88.98%
Pointnet^{L}	2.8M	81.37%	81.25%	81.49%	83.65%	76.25%	78.83%	80.5%	72.1%	85.06%	86.8%
Point-Baseline^{L}	0.5M	60.74%	65.55%	54.38%	62.44%	56.6%	56.13%	71.62%	56.43%	66.24%	47.77%
DeiT-base^{*I}	85.7M	82.19%	82.5%	81.87%	82.29%	**82.35**%	78.1%	81.22%	**74.45**%	79.61%	82.72%
DeiT-tiny^{*I}	5.7M	80.01%	80.94%	78.99%	80.03%	80.23%	76.89%	80.53%	73.87%	78.3%	76.99%
DeiT-tiny^{I}	5.7M	80.77%	81.46%	80.03%	80.99%	80.85%	77.12%	74.38%	73.54%	76.16%	77.27%
DGCNN^{L}	0.6M	80.82%	80.38%	81.24%	83.44%	75.24%	**78.59**%	79.17%	71.01%	85.75%	83.03%
Point-Transformer^{L}	0.5M	**83.18**%	**82.89**%	**83.45**%	**85.43**%	78.56%	76.13%	**83.4**%	72.5%	**86.73**%	**85.1**%
Pointnet^{L}	2.8M	80.2%	79.84%	80.55%	82.76%	74.84%	78.19%	80.6%	68.09%	83.43%	82.73%
Point-Baseline^{L}	0.5M	60.23%	63.64%	56.13%	61.76%	56.74%	56.2%	66.98%	58.11%	65.67%	52.25%

*: Pre-trained & fine-tuned on ImageNet 1k, ^{I}: using RGB data, ^{L}: using LiDAR data

Table 5.2: **Point-based v.s. image-based object ReID.** The table shows object ReID results for point models and image models using our proposed RTMM matching head. Models in the top section are trained on nuScenes and evaluated on *nuScenes eval*, while models in the bottom two sections are trained on WOD and evaluated on *Waymo eval 1* and *Waymo eval 2*, respectively. *Waymo eval 1* filters out observations that do not have an associated image crop, while *Waymo eval 2* does not. All evaluation sets filter out observations with fewer than two points.

image backbone is roughly half as efficient as the corresponding point models. Point-Transfromer2M, the scaled-up model used in our scaling experiments runs at a similar efficiency to DeiT-Tiny. When combined with RTMM, all models except for DeiT-Base and Point-Transfromer7M run in real-time (faster than 10 Hz).

Training Details All models of the same modality in table 5.2 were trained using identical hyperparameters and their final checkpoints are used for evaluation. We used the AdamW [77] optimizer with a learning rate of $1e-5$ (images) and $3e-4$ (point clouds), weight decay of 0.01, and cosine learning rate and momentum schedules [122]. We use a batch size of 64×4 GPUs (images) and 256×4 GPUs (point clouds) and note that our batch normalization layers were not synchronized across devices during training. We also note that training using the learning rate schedule of the other modality decreases final validation accuracy for all models. Pre-trained models are trained for 200 epochs each, while randomly initialized models are optimized for 500 epochs and 400 epochs on nuScenes and WOD, respectively. The epoch numbers for randomly initialized models were not chosen arbitrarily; they roughly correspond to the same number of gradient descent steps (± 3 epochs) across both datasets as the Waymo dataset is larger.

5.3 Experiments

Our empirical evaluation is based on two re-identification datasets created from nuScenes and WOD (details provided in sec.4). The difference in LiDAR resolution between each dataset (32 v.s. 64 beam, respectively), allows us to establish how ReID performance varies as sensor resolution increases. We also establish the relative performance of image-based and point-based ReID, how performance varies with respect to point density within a dataset, that scaling training time significantly increases our models' performance, and that our models are capable of correcting the errors of strong motion-based multi-object trackers. Finally, we provide an ablation study to assess whether common representation learning techniques, such as triplet loss [46], a completion task [32, 50], or simply adding a classification head, can help to improve the performance of point ReID networks.

5.3.1 A Comparison between point-based and image-based ReID

Table. 5.2 reports the results of our large-scale empirical study. The top section of the table corresponds to models trained on nuScenes and evaluated on *nuScenes eval*. The bottom two sections correspond to models trained on Waymo and evaluated on *Waymo eval 1* and *Waymo eval 2*, respectively. Matching accuracy is reported overall and for each individual class. We also report F1 scores for positive and negative matches. This experimental study has three main goals: to measure point ReID performance as sensor resolution is increased, to establish the performance difference between point-based and image-based ReID networks, and to measure point ReID performance as point density is increased within a dataset.

When comparing the accuracy of models trained on nuScenes to those trained on WOD, we observe that there is an overall increase for all models except the point baseline. However, the point models improve by a much greater margin than image models: as much as 11.92% for the point transformer versus a 5.41% increase for the randomly initialized DeiT-tiny model. We hypothesize that the performance increase of image models is due to the following reasons: 1) the image sensors are of higher resolution on WOD and 2) the WOD training set is much more diverse—it has 80% more objects. This second reason is a potential confounder when assessing the extent to which the increase in point density improves point ReID performance. However, the smaller relative increase for image models allows us to account for the potentially confounding effect of a more diverse training set on WOD, showing that the increase in sensor resolution from nuScenes to WOD causes a performance improvement of at least a $11.92\% - 5.41\% = 6.51\%$ for our point ReID models. The story is very different, however, for the baseline model. We hypothesize that it fails to learn any meaningful representations for matching, which is why improvements from increased sensor resolution or dataset diversity have no impact on its performance. This shows that simply concatenating the pooled representations of the point backbone is insufficient for the tracking-by-detection matching task. All our models on both datasets learn an unbiased matching function on aggregate as is evidenced by similar positive and negative F1 scores. When looking at class-specific results, we note that all models follow a similar increase from nuScenes to WOD as can be observed for accuracy, except for some image models whose performance decreases on the Bus class. Of all classes, pedestrian and bicycle benefit the most from the increase in LiDAR sensor resolution with respective increases of 18.01% and 14.67%. This is a boon for our models' applicability to 3D MOT as pedestrians and cyclists are some of the most likely objects to become occluded and can easily be lost by simple motion-model-based trackers. We note that truck and bus benefit the least from increasing LiDAR sensor resolution. We hypothesize that this is because large objects will have many points regardless of the sensor's resolution.

Comparing the point re-identification models, Point-Transformer performs best on WOD, while all models perform very similarly on the nuScenes dataset. Focusing on image models exclusively, we note that the pre-trained DeiT-Base model performs best of all as is expected given its large number of parameters. Directly comparing point models to image models, we observe that image models always outperform their point counterparts when observations are visible to both camera and LiDAR sensors, but that increasing sensor resolution considerably decreases this gap. When comparing the Point-Transformer to the randomly initialized DeiT-Tiny on WOD, we observe the smallest performance gap between large rigid objects (bus, truck, and car), while the smaller deformable objects (pedestrian and bicycle) pose more difficulty to the Point Transformer. This is to be expected as deformable objects create inherent shape ambiguity, which can be resolved in images by leveraging color or texture information, but for point clouds, an object's shape is its primary distinguishing characteristic. While the point models perform poorer than image models overall, the relative improvement seen from nuScenes to WOD is non-trivial and suggests that the gap in performance will shrink as LiDAR sensor resolution continues

Backbone	Epoch	Acc.	F1 Pos.	F1 Neg.	Car	Pedestrian	Bicycle	Bus	Motorcycle	Truck	FP
Point-Transformer2M	3200	**87.48%**	**87.58%**	**87.38%**	**89.24%**	**83.56%**	**83.66%**	87.03%	**80.75%**	**90.5%**	90.36%
Point-Transformer2M	1600	87.16%	87.15%	87.16%	89.18%	82.68%	82.25%	**87.19%**	78.57%	90.33%	**91.25%**
Point-Transformer2M	800	86.14%	86.14%	86.13%	88.17%	81.62%	81.49%	86.64%	76.77%	89.76%	90.34%
Point-Transformer2M	400	84.5%	84.57%	84.43%	86.64%	79.82%	78.56%	85.38%	73.03%	87.23%	88.3%
Point-Transformer2M	200	82.56%	82.67%	82.46%	84.9%	77.45%	76.11%	84.43%	70.69%	85.66%	86.53%

Table 5.3: **Scaling training time significantly improves performance on *waymo eval 1*.**

to increase.

Intra-dataset point density comparison Figures 5.2 and 5.3 graph the accuracy of our models when evaluated at different point densities within a dataset. The goal of this stratified evaluation is threefold: to show that point ReID performance approaches image ReID performance at higher point densities, to showcase the best possible performance one can expect from our point ReID models, and to give an idea of how performance might improve when even denser LiDAR sensors become available. We note that LiDAR sensors with up to 128 beams (2 times the resolution of WOD's 64-beam LiDAR) are already commercially available today [124].

Figure 5.2 plots the overall point cloud ReID performance on *nuScenes eval* (left) and *Waymo eval 1* (right). Each plot graphs the accuracy of every model (except the baseline) on pairs of point cloud observations $(\{\boldsymbol{x}_1^{(i)}\}_i^{n_1}, \{\boldsymbol{x}_2^{(i)}\}_i^{n_2})$, where #points $\leq \min(n_1, n_2)$; that is pairs that both have more than x points. We chose to exclude the baseline as its poor performance makes all lines look flat. The number of positive and negative examples for each threshold is shown below the x-axis. We observe that our point models' accuracy increases with the number of points approaching or surpassing image model performance at the highest densities, while the image models remain relatively flat throughout. We note that the nuScenes image models do experience a slight decrease in performance at higher point densities (more details are provided in sec.6.1). On nuScenes, all point models perform relatively similarly throughout, but on WOD we observe the stronger performance of the Point-Transformer at lower point densities, while PointNet performs the best at higher densities.

Figure 5.3 plots the performance of our ReID models on *Waymo eval 1* for specific point density power-two buckets ($[2^i, 2^{i+1})$). Each individual plot graphs matching accuracy on pairs of observations where one element of the pair belongs to the bucket indicated in the plot's title and the other element belongs to the bucket on the x-axis. Instead of showing cumulative accuracy numbers as reported in fig. 5.2 these plots report more precise values. The plots show a convincing story: when comparing pairs of observations,

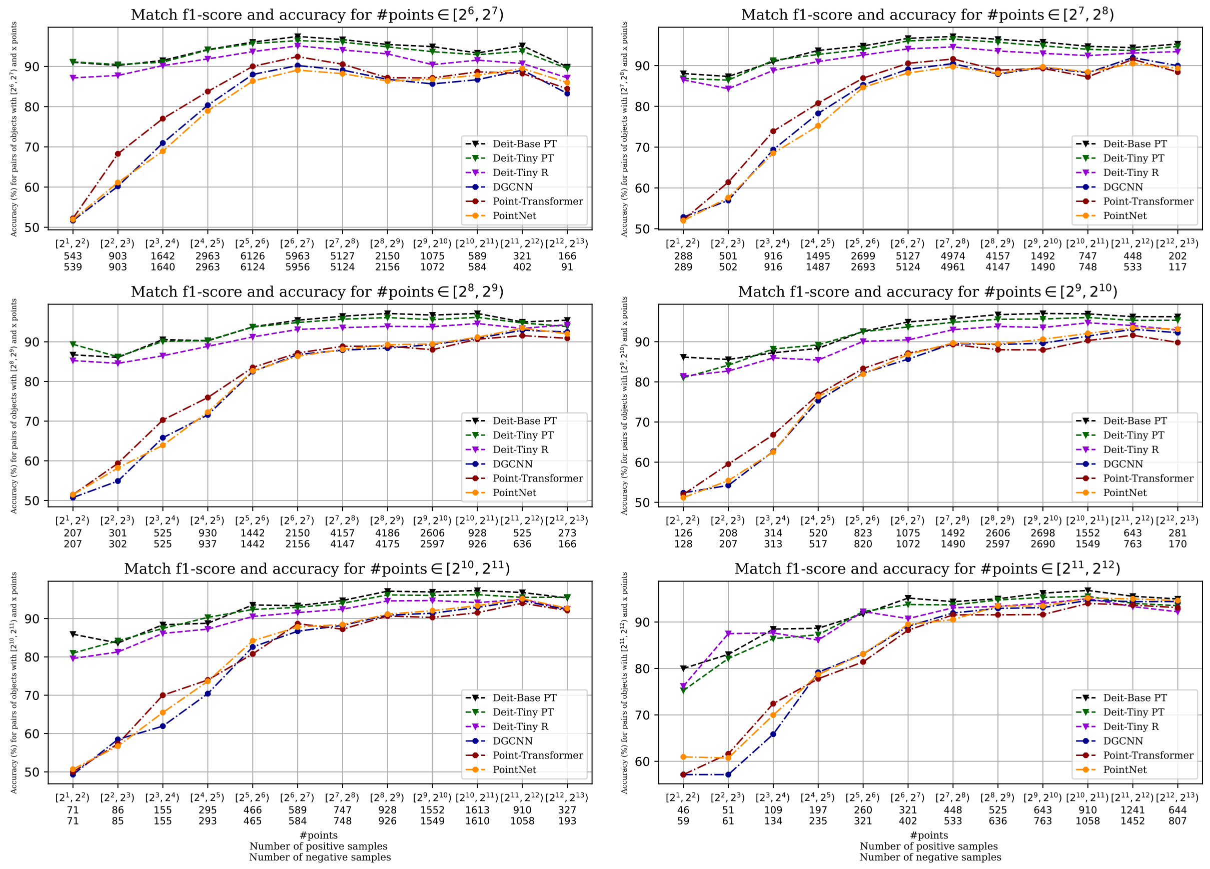

Figure 5.3: **Point cloud ReID matches image ReID performance when observations are sufficiently dense.** The plots show the performance of our ReID models on *Waymo eval 1* for specific point density power-two buckets ($[2^i, 2^{i+1})$). Each individual plot graphs matching accuracy on pairs of observations where one element of the pair belongs to the bucket indicated in the plot's title and the other element belongs to the bucket on the x-axis. We observe that performance stabilizes when both objects in the pairs have more than 64 points.

if each observation in the pair has more than 64 points, then ReID accuracy remains stable at or above 88%. Furthermore, performance continues to increase as the points in each observation are increased. This can be observed despite the inputs to all our models being subsampled to 128 points. At the highest point densities, we see the curves reach those of the image models. This shows that point ReID models can currently approach image-level performance, a trend that will be amplified in the future with increases in LiDAR sensor resolution.

5.4 Scaling point re-identification

As illustrated in table 4.3 the number of samples in our ReID datasets is combinatorially large. For WOD, there are $4.35e8$ positive samples and $3.89e19$ negative samples. Since each training epoch samples one positive or one negative sample with equal probability for each object in the training set, there will be approximately $31,878$ positive pairs and $31,878$ negative pairs sampled every epoch on WOD. At this rate, it would take $\sim 13,646$ epochs to sample all possible positive samples on the WOD ReID dataset. Since we only train our models for 400 epochs ($\sim$ 9 hours on four Nvidia A6000 GPUs), it is quite likely that we are underfitting the training set. In an effort to determine the importance of training for longer, we train five models for $200 \cdot 2^i$ epochs with $i \in \{0, 1, 2, 3, 4\}$ on the WOD. For these experiments, we use a slightly larger 2 million parameter Point-Transformer model as it has more capacity than the 500k parameter model used in our previous experiments, but still has inference time on par with DeiT-Tiny. We also used a slightly larger batch size for these experiments 256×5 GPUs keeping the learning rate and other hyperparameters fixed. Table 5.3 shows the performance of Point-Transformer2M models trained for different numbers of epochs. As expected, the model trained for the longest (3200 epochs) achieves the strongest overall accuracy and the best or close to the best accuracy for every individual class. Of all classes, the motorcycle class sees the largest improvement from increased training time: +10.06%. This is expected as it has the least number of samples in the training set and is therefore sampled the least at training time. Additionally, we observe that the performance improvement of training longer seems to start saturating between the models trained for $1600e$ and $3200e$. While these numbers can be noisy and more trials are needed to obtain reliable estimates, this suggests that performance gains may be limited for further training.

Figure 5.4 plots the accuracy of the scaled models on *waymo eval 1* as a function of the point density. We observe that the increased training time leads to improved performance at every point density. Additionally, it appears that performance at smaller point densities

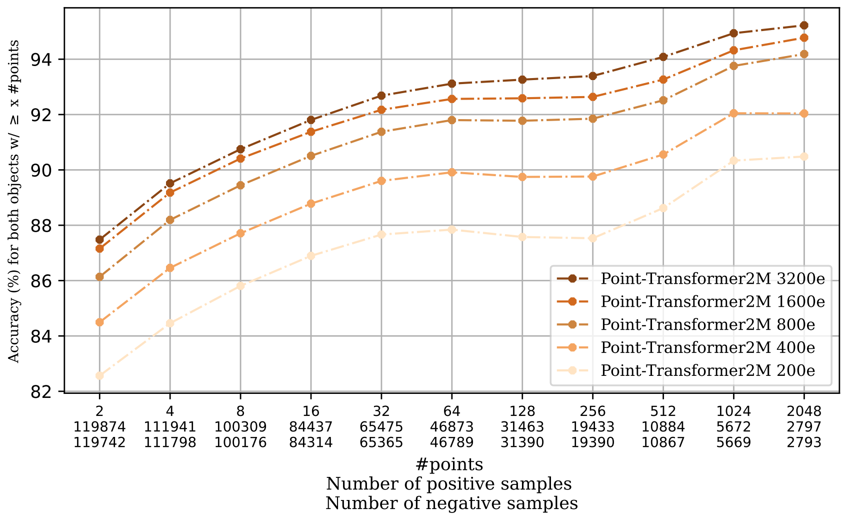

Figure 5.4: **Training for longer leads to a significant increase in accuracy.** The plot shows performance as a function of point density. We see that models trained for longer (darker) perform significantly better.

saturates faster as scale increases than with larger point densities. We hypothesize that this is due to two reasons: 1) for very few points, even with more training time, it is impossible to learn something without enough information and 2) observations at smaller point densities are sampled more frequently during training because they are more prevalent. This suggests that a sampling algorithm that explicitly balances samples at different point densities may be beneficial.

Figure 5.5 plots the performance of our ReID models on *Waymo eval 1* for specific point density power-two buckets ($[2^i, 2^{i+1})$). Each individual plot graphs matching accuracy on pairs of observations where one element of the pair belongs to the bucket indicated in the plot's title and the other element belongs to the bucket on the x-axis. Instead of showing cumulative accuracy numbers as reported in fig. 5.4 these plots give more precise values. We observe similar trends to figure 5.3 where ReID accuracy remains stable for pairs with two observations at or above 88%. Additionally, we see that the model trained for 3200 epochs reaches the performance of the randomly initialized DeiT-Tiny model much earlier on and even surpasses it at the highest point density. This shows that considerable performance improvements can be made by training for longer.

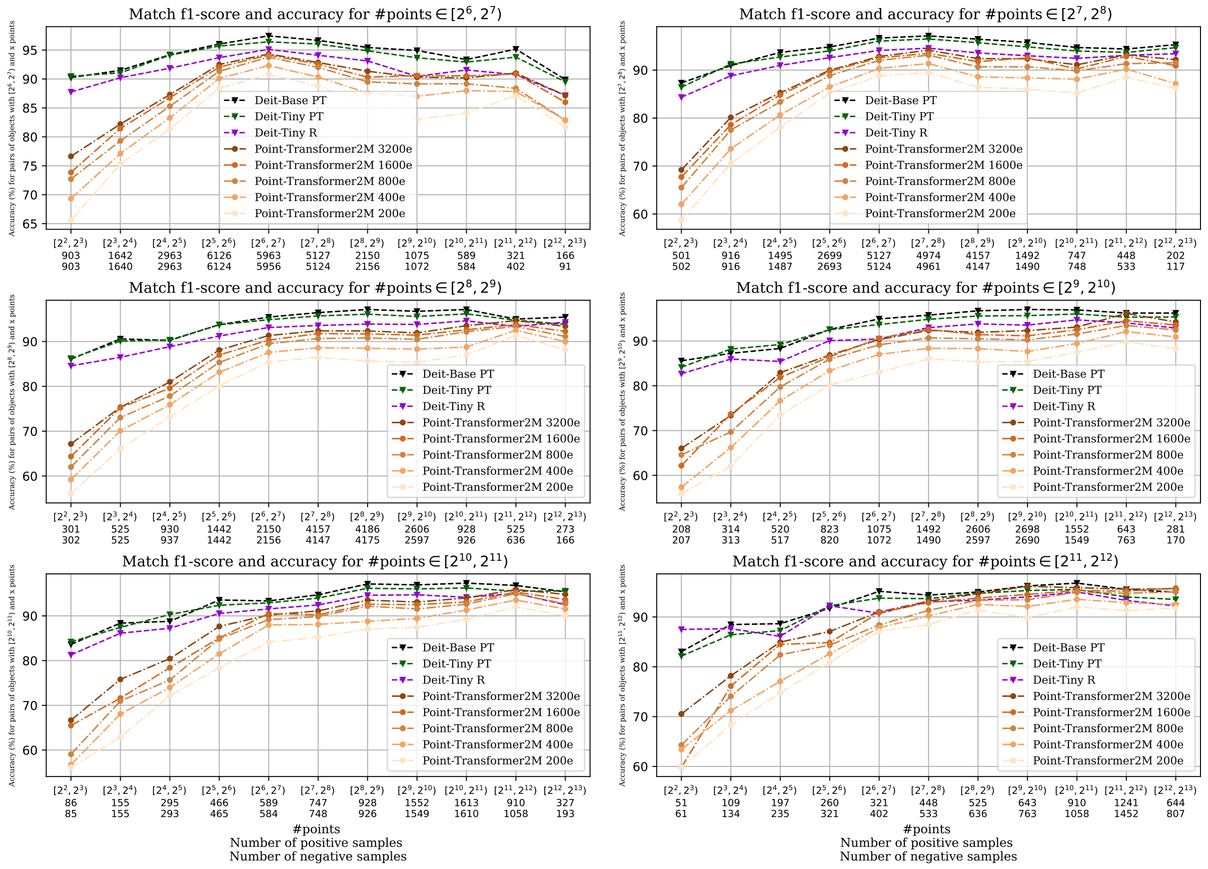

Figure 5.5: **Scaled point ReID models can surpass the performance of image ReID for dense enough observations.** The plots show the performance of our ReID models on *Waymo eval 1* for specific point density power-two buckets ($[2^i, 2^{i+1})$). Each individual plot graphs matching accuracy on pairs of observations where one element of the pair belongs to the bucket indicated in the plot's title and the other element belongs to the bucket on the x-axis. We observe that performance stabilizes when both objects in the pairs have more than 64 points. Additionally, we note that the 3200 epoch scaled model attains very competitive performance with DeiT-Tiny R.

5.5 Correcting tracking errors

To demonstrate that our Re-ID networks can benefit 3D MOT, we extract errors from 3 strong multi-object trackers on the nuScenes validation set: Probabilistic MOT [15], Simple Track [98], and a differentiable tracker that we implemented ourselves (further details of the trackers are provided in the appendix as well as their full tracking scores, see table. A.1). This is done by extracting instances where the tracker incorrectly matched one ground truth object to another (negative pairs), or situations where the tracker failed to match a ground truth object from one frame to the next (positive pairs). Since nuScenes provides object ids, we can easily find the corresponding detections in our ReID dataset and verify the performance of our point ReID models on them. This works because we use the same BEVfusion C+L detections are used as input to our trackers and for creating the nuScenes ReID dataset.

Specifically, to extract positive examples from tracking errors, we collect pairs of detections matched to a ground truth track where the predicted tracks that previously tracked a ground truth object (GT_1 at timestep T) incorrectly switch (IDS) to another object or simply miss the prediction at timestep $T+k$ (fragmentation), despite GT_1 still being detected. The positive pair then becomes the detection matched to GT_1 at timestep T and the true positive detection for GT_1 at timestep $T+k$. To extract negative examples, we collect predicted tracks that switch between two GT tracks (GT_1 at T and GT_2 at $T+k$). The negative pair then becomes the true positive detection for GT_1 at T and the true positive detection for GT_2 at $T+k$.

Figure 5.6 plots the performance of our ReID networks on mistakes made by different trackers. We observe similar accuracy to what is reported in table 5.2 for all models; image ReID models perform the best, followed by point ReID models and finally the baseline performs worst of all. This shows that point ReID models can be useful in a 3D MOT context. Given that this performance was predictable from the results of table 5.2 we hypothesize that the performance improvements from nuScenes to WOD will also port to the errors of 3D multi-object tracker. That is, point ReID models will become even more useful to 3D MOT as sensor resolution increases.

5.6 Ablation study

To provide more insight into our choice of training objective (binary cross-entropy (BCE)), we provide an ablation study showing different losses we tried. In addition to BCE, we

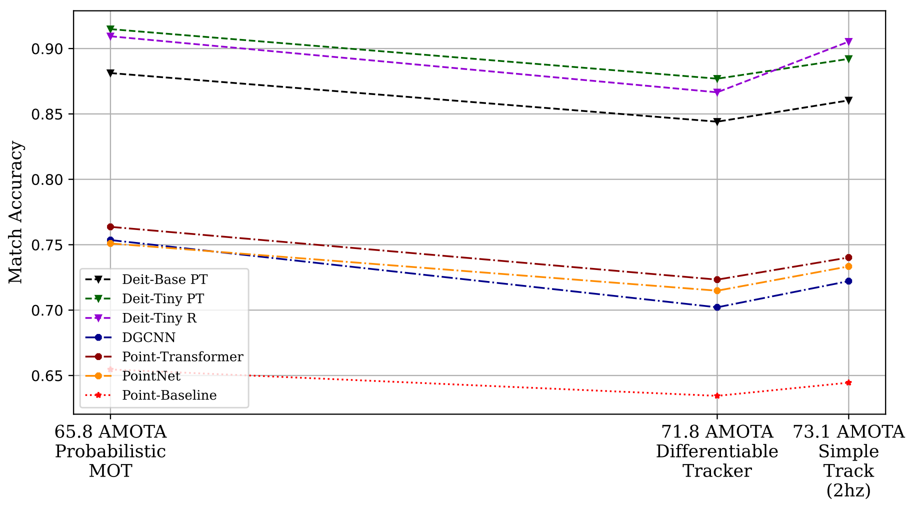

Figure 5.6: **Point cloud ReID networks can correct a motion-based tracker's errors.** The plot shows ReID accuracy as a function of the AMOTA score of the tracker used to extract the errors.

experimented with triplet loss[46] $\mathcal{L}_{triplet}$ as used by [172, 31, 166], a reconstruction objective $\mathcal{L}_{shape}$ as in [32, 50, 144], and a simple classification objective $\mathcal{L}_{cls}$. In what follows, we described the loss formulations added to our pipeline in figure 5.7 and discuss how performance varies when they are combined with BCE.

To extract triplets from training batches sampled according to algorithm 1, we design a triplet sampling algorithm. It works as follows: for each positive pair of samples in the batch, we sample 128 associated negative examples (n) uniformly at random from the batch. We use the first element of the positive pair as the anchor, a, the second element is the positive sample p. Since we use an effective batch size of $4 \times 256 = 1024$ to train our networks and approximately half of the samples drawn from algorithm 1 are positive, each batch fed to the triplet loss will have $M =\sim 65536$ samples. To compute the triplet loss, the point cloud representation $f_\theta(\boldsymbol{X}) \in \mathbb{R}^{64\times128}$ is flattened yielding stacked anchor, positive, and negative batches: $\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{N} \in \mathbb{R}^{\sim 65536\times8192}$. Then, following loss formulation is used

$$\mathcal{L}_{triplet}(\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{N}) = \frac{1}{M}\sum_{i=0}^{M} \max\left(0, m + \|\boldsymbol{A}_{i,:} - \boldsymbol{P}_{i,:}\|_2 - \|\boldsymbol{A}_{i,:} - \boldsymbol{N}_{i,:}\|_2\right), \tag{5.8}$$

where $m = 10$ and i indexes the rows of $\boldsymbol{A}, \boldsymbol{P}$, and $\boldsymbol{N}$. While this formulation is the one that we show in table 5.4, we also experimented with applying a pooling operation over the

$\mathcal{L}_{match}$	$\mathcal{L}_{triplet}$	$\mathcal{L}_{shape}$	$\mathcal{L}_{cls}$	Acc.	F1 Pos.	F1 Neg.	Car	Pedestrian	Bicycle	Bus	Motorcycle	Truck	FP
✓				84.46%	84.47%	84.44%	86.76%	79.37%	78.56%	86.24%	73.11%	88.02%	88.65%
✓	✓			84.44%	84.46%	84.43%	86.75%	79.47%	78.28%	84.83%	71.71%	86.4%	88.6%
✓		✓		84.73%	84.72%	84.75%	87%	79.7%	78.83%	85.3%	75.84%	88.45%	89.08%
✓			✓	84.13%	84.29%	83.97%	86.24%	79.42%	79.42%	84.43%	75.45%	87.88%	88.78%
✓	✓	✓		84.74%	84.75%	84.74%	87.11%	79.67%	78.77%	83.49%	70.54%	87.14%	88.94%
✓		✓	✓	84.08%	84.24%	83.92%	86.22%	79.29%	79.75%	84.2%	74.98%	88.31%	88.73%
✓	✓		✓	84.96%	84.96%	84.96%	87.29%	79.96%	79.53%	83.65%	71.16%	87.28%	89.63%
✓	✓	✓	✓	85.19%	85.19%	85.19%	87.46%	80.29%	79.42%	83.88%	72.17%	88.17%	89.87%

Table 5.4: **Ablation study for different losses.** Each row in the table represents a distinct Point-Transformer model trained on WOD and evaluated on *Waymo eval 1*.

set dimensions of $f_\theta(\boldsymbol{X})$ before creating $\boldsymbol{A}, \boldsymbol{P}$, and $\boldsymbol{N}$ and with using the representations computed from RTMM. The former performed worse than the proposed formulation, while the latter was trivially minimized and achieved 0 loss shortly after the start of training.

We also explore adding a shape completion objective similar to those used by [32, 50]. To compute $\mathcal{L}_{shape}$ we use the Chamfer distance between an upsampled version of the input point cloud, $\hat{\boldsymbol{y}}_{shape}$, and the ground truth $\boldsymbol{y}_{shape}$ created by subsampling a dense version (see sec.4.2) of the input sample to 2048 points. We note that the loss is not computed for false positives as they have no corresponding dense objects. We compute $\hat{\boldsymbol{y}}_{shape} = \text{Conv1D}_{us}(f_\theta(\boldsymbol{X}))$, where Conv1D_{us} is a series of 1D convolution blocks (Conv1D,Norm,ReLU) which converts $f_\theta(\boldsymbol{X}) \in \mathbb{R}^{64\times128}$ to $\hat{\boldsymbol{y}}_{shape} \in \mathbb{R}^{3\times2048}$. The loss is computed as follows:

$$\mathcal{L}_{shape}(Q, P) = \sum_{p\in P} \min_{q\in Q} \|p - q\|_2^2 + \sum_{q\in Q} \min_{p\in P} \|q - p\|_2^2, \tag{5.9}$$

where $Q = \hat{\boldsymbol{y}}_{shape}$ and $P = \boldsymbol{y}_{shape}$ in our experiments.

The last objective we experiment with is categorical cross-entropy. Specifically, we train the network to predict the class of each object from the pooled hidden representation $\text{pool}(f_\theta(\boldsymbol{X}))$ output by the point backbone. For false positives, we create extra classes corresponding to each possible misclassification. This effectively doubles the number of classes (C). The loss is computed as follows

$$\mathcal{L}_{cls}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{2\cdot C} \boldsymbol{y}_i \cdot \log(\hat{\boldsymbol{y}}_i), \tag{5.10}$$

where $\hat{\boldsymbol{y}} = softmax(\text{MLP}_{cls}(\text{pool}(f_\theta(\boldsymbol{X}))))$.

Table 5.4 shows the performance of Point-Transformer models trained with combinations of the different objectives on WOD. We use the same hyperparameters as described

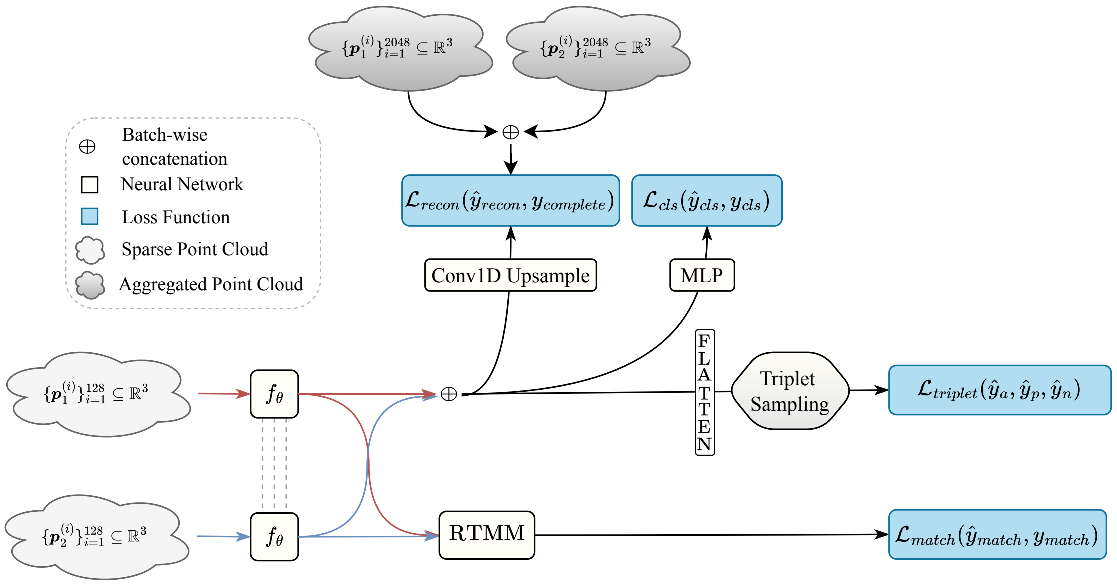
$\{p_1^{(i)}\}_{i=1}^{2048} \subseteq \mathbb{R}^3$
$\{p_2^{(i)}\}_{i=1}^{2048} \subseteq \mathbb{R}^3$
Batch-wise concatenation
Neural Network
Loss Function
Sparse Point Cloud
Aggregated Point Cloud
$\mathcal{L}_{recon}(\hat{y}_{recon}, y_{complete})$
$\mathcal{L}_{cls}(\hat{y}_{cls}, y_{cls})$
Conv1D Upsample
MLP
FLATTEN
Triplet Sampling
$\mathcal{L}_{triplet}(\hat{y}_a, \hat{y}_p, \hat{y}_n)$
$\{p_1^{(i)}\}_{i=1}^{128} \subseteq \mathbb{R}^3$
$\{p_2^{(i)}\}_{i=1}^{128} \subseteq \mathbb{R}^3$
f_θ
f_θ
RTMM
$\mathcal{L}_{match}(\hat{y}_{match}, y_{match})$

Figure 5.7: **Network diagram for the point cloud ReID networks in our ablation study.** The diagram depicts the flow from input to loss functions for the different objectives used in our ablations study.

in section 5.2. All models obtain a final validation accuracy between 84.08% and 85.19%, showing largely inconclusive results which could have been the work of regular randomness associated with non-deterministic GPU operations. Given the lack of empirical evidence of a stronger performing formulation, we chose to train our models using only the binary cross-entropy objective.

Chapter 6

Limitations, Conclusion, and Future Work

6.1 Limitations

This thesis provides an empirical evaluation of point cloud ReID for multi-object tracking, however, some limitations affect the impact of the work. The most glaring limitation is the proposed training-time sampling algorithm. Although algorithm 1 might seem intuitive, further inspection reveals that it introduces a bias towards positive samples at higher point densities (see figure A.1). This would go unnoticed if the same sampling algorithm were used at test time. However, we opted to use a balanced sampling algorithm at testing time to expose any potential biases learned during training. This limitation impacts our results as our models inherit the sampling algorithm's bias during training. This can be observed in the left side of figure 5.2 (image models) and on the top two plots of figure A.1 (point and image models). Thankfully, however, the bias does not affect the soundness of our results, other than the reported point cloud ReID accuracy being lower than it could have otherwise been. We defer further discussion of this limitation to section A.3.1 of the appendix. Other limitations include the poorly performing point cloud ReID baseline, a lack of experiments elucidating the effect of sensor noise on performance, and missing results for tracking errors on WOD. While the point baseline's comparison mechanism—concatenating pooled backbone representations—is certainly reasonable, its extremely poor performance limits the model's ability to be an effective baseline. Selecting a stronger baseline algorithm would help remedy this flaw and highlight the strength of RTMM. The problem of bounding box noise certainly needs to be addressed by any effective ReID network operating in a tracking-by-detection context. While we provide extensive performance evaluation of our ReID models, the effect that sensor noise has on performance is not accounted for. Additional experiments using an analogous dataset generated from ground truth bounding boxes could help answer this question. Finally, experiments re-identifying tracking errors from the WOD tracking benchmark would improve the impact of the work.

6.2 Conclusion

We have conducted the first large-scale study of object re-identification from cropped point cloud observations in a 3D multi-object tracking context. Our findings can be summarized as follows: 1) we establish the performance of point cloud ReID relative to image ReID, 2) we show that our point ReID networks can attain strong ReID performance, even on par with image models, as long as the compared observations are sufficiently dense, 3) we established that point ReID performance increases as LiDAR sensor resolution is increased, 4) we demonstrated the performance of point ReID models can be substantially increased by training for longer, and 5) we demonstrated that our point ReID networks can re-identify pairs of objects incorrectly tracked by three strong multi-object trackers—demonstrating point ReID's potential to enhance 3D MOT.

While image ReID outperforms point ReID when observations are visible to both sensors, our results show that the latter still attains strong enough performance to be useful in a 3D MOT pipeline. Therefore, methods should be developed to leverage this newly discovered capability. For the time being, autonomous driving systems like the WOD vehicle, which have limited camera FOV, stand to benefit the most from the added complementarity of a ReID network processing 360° LiDAR scans. However, even vehicles equipped with cameras covering 360° can benefit from the added redundancy of point ReID, especially in cases where the observations are sufficiently dense to be reliable. In the future, as LiDAR technology continues to advance, point ReID performance can only increase—magnifying the implications of our findings. Already today, bleeding edge LiDAR sensors feature 128 beams[124], twice the resolution of WOD's top-mounted LiDAR.

6.3 Future Work

Our initial study of ReID from point clouds opens many directions for future work. Integrating our point ReID models into a multi-object tracker is a logical next step. Another possible direction is to investigate multi-modal object ReID in a multi-object tracking-by-detection context; using techniques such as [88, 136] could work well for combining point clouds and images, especially in our sequence-based framework. Another interesting direction would be to study how geometric priors, such as SE(3) or SO(3) equivariance [21, 27], can enhance ReID from point clouds (e.g. to become invariant to the detector's noise). Finally, figure A.1 shows that our training time sampling algorithm is biased at higher point densities; therefore, an investigation into better sampling algorithms is in order.

References

[1] Serge Assaad, Carlton Downey, Rami Al-Rfou, Nigamaa Nayakanti, and Ben Sapp. Vn-transformer: Rotation-equivariant attention for vector neurons. *CoRR*, abs/2206.04176, 2022.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Seung Hwan Bae and Kuk-Jin Yoon. Robust online multiobject tracking with data association and track management. *IEEE Trans. Image Process.*, 23(7):2820–2833, 2014.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, pages 737–744. Morgan Kaufmann, 1993.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio

Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[7] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11618–11628. Computer Vision Foundation / IEEE, 2020.

[8] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2020.

[10] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] Jiaxing Chen, Xinyang Jiang, Fudong Wang, Jun Zhang, Feng Zheng, Xing Sun, and Wei-Shi Zheng. Learning 3d shape feature for texture-insensitive person reidentification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 8146–8155. Computer Vision Foundation / IEEE, 2021.

[12] Ke Chen and Ahmad Salman. Extracting speaker-specific information with a regularized siamese deep network. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 298–306, 2011.

[13] Long Chen, Haizhou Ai, Chong Shang, Zijie Zhuang, and Bo Bai. Online multi-object tracking with convolutional neural networks. In *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*, pages 645–649. IEEE, 2017.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[15] Hsu-Kuang Chiu, Jie Li, Rares Ambrus, and Jeannette Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 14227–14233. IEEE, 2021.

[16] Jaesung Choe, Chunghyun Park, François Rameau, Jaesik Park, and In So Kweon. Pointmixer: Mlp-mixer for point cloud understanding. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVII*, volume 13687 of *Lecture Notes in Computer Science*, pages 620–640. Springer, 2022.

[17] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4846–4855. IEEE Computer Society, 2017.

[18] Dahjung Chung, Khalid Tahboub, and Edward J. Delp. A two stream siamese convolutional neural network for person re-identification. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1992–2000. IEEE Computer Society, 2017.

[19] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020.

[20] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao

Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. *CoRR*, abs/2302.05442, 2023.

[21] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12180–12189. IEEE, 2021.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[24] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[25] Viktor Eckstein, Arne Schumann, and Andreas Specker. Large scale vehicle reidentification by knowledge transfer from simulated data and temporal attention. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2626–2631. Computer Vision Foundation / IEEE, 2020.

[26] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021.

[27] Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In Hugo Larochelle,

Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[28] Cunyuan Gao, Yi Hu, Yi Zhang, Rui Yao, Yong Zhou, and Jiaqi Zhao. Vehicle re-identification based on complementary features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2520–2526. Computer Vision Foundation / IEEE, 2020.

[29] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *CoRR*, abs/2006.12671, 2020.

[30] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[31] Adhiraj Ghosh, Kuruparan Shanmugalingam, and Wen-Yan Lin. Relation preserving triplet mining for stabilising the triplet loss in re-identification systems. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*, pages 4829–4838. IEEE, 2023.

[32] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 1359–1368. Computer Vision Foundation / IEEE, 2019.

[33] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[34] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[35] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*, pages 10335–10342. IEEE, 2020.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[37] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

[39] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In Helen Meng, Bo Xu, and Thomas Fang Zheng, editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 5036–5040. ISCA, 2020.

[40] JunYoung Gwak, Silvio Savarese, and Jeannette Bohg. Minkowski tracker: A sparse spatio-temporal R-CNN for joint object detection and tracking. *CoRR*, abs/2208.10056, 2022.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[42] Shuting He, Hao Luo, Weihua Chen, Miao Zhang, Yuqi Zhang, Fan Wang, Hao Li, and Wei Jiang. Multi-domain learning and identity mining for vehicle re-identification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2485–2493. Computer Vision Foundation / IEEE, 2020.

[43] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 14993–15002. IEEE, 2021.

[44] Heisner. The unacceptability of dreams.

[45] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Pointgmm: A neural GMM network for point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 12051–12060. Computer Vision Foundation / IEEE, 2020.

[46] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[47] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

[48] Zheng Hu, Chuang Zhu, and Gang He. Hard-sample guided hybrid contrast learning for unsupervised person re-identification. In *7th IEEE International Conference on Network Intelligence and Digital Content, IC-NIDC 2021, Beijing, China, November 17-19, 2021*, pages 91–95. IEEE, 2021.

[49] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnet: Enhancing point features with image semantics for 3d object detection. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 35–52, Cham, 2020. Springer International Publishing.

[50] Le Hui, Lingpeng Wang, Mingmei Cheng, Jin Xie, and Jian Yang. 3d siamese voxel-to-bev tracker for sparse point clouds. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 28714–28727, 2021.

[51] Le Hui, Lingpeng Wang, Linghua Tang, Kaihao Lan, Jin Xie, and Jian Yang. 3d siamese transformer network for single object tracking on point clouds. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel*

Aviv, Israel, October 23-27, 2022, Proceedings, Part II, volume 13662 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2022.

[52] Bingliang Jiao, Lingqiao Liu, Liying Gao, Guosheng Lin, Lu Yang, Shizhou Zhang, Peng Wang, and Yanning Zhang. Dynamically transformed instance normalization network for generalizable person re-identification. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XIV*, volume 13674 of *Lecture Notes in Computer Science*, pages 285–301. Springer, 2022.

[53] Xin Jin, Tianyu He, Kecheng Zheng, Zhiheng Yin, Xu Shen, Zhen Huang, Ruoyu Feng, Jianqiang Huang, Zhibo Chen, and Xian-Sheng Hua. Cloth-changing person re-identification from A single image with gait prediction and regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 14258–14267. IEEE, 2022.

[54] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[55] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 2020.

[56] Pirazh Khorramshahi, Neehar Peri, Jun-Cheng Chen, and Rama Chellappa. The devil is in the details: Self-supervised attention for vehicle re-identification. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIV*, volume 12359 of *Lecture Notes in Computer Science*, pages 369–386. Springer, 2020.

[57] Aleksandr Kim, Guillem Brasó, Aljosa Osep, and Laura Leal-Taixé. Polarmot: How far can geometric relations take us in 3d multi-object tracking? In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel,*

October 23-27, 2022, Proceedings, Part XXII, volume 13682 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2022.

[58] Aleksandr Kim, Aljosa Osep, and Laura Leal-Taixé. Eagermot: 3d multi-object tracking via sensor fusion. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 11315–11321. IEEE, 2021.

[59] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[60] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation, 2017.

[61] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[62] Eric-Tuan Le, Iasonas Kokkinos, and Niloy J. Mitra. Going deeper with lean point networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9500–9509. Computer Vision Foundation / IEEE, 2020.

[63] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese CNN for robust target association. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016*, pages 418–425. IEEE Computer Society, 2016.

[64] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[65] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[66] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11550–11559. Computer Vision Foundation / IEEE, 2020.

[67] Mingchao Liang and Florian Meyer. Neural enhanced belief propagation for data association in multiobject tracking. In *25th International Conference on Information Fusion, FUSION 2022, Linköping, Sweden, July 4-7, 2022*, pages 1–7. IEEE, 2022.

[68] Daniele Liciotti, Marina Paolanti, Emanuele Frontoni, Adriano Mancini, and Primo Zingaretti. Person re-identification dataset with RGB-D camera in a top-view configuration. In Kamal Nasrollahi, Cosimo Distante, Gang Hua, Andrea Cavallaro, Thomas B. Moeslund, Sebastiano Battiato, and Qiang Ji, editors, *Video Analytics. Face and Facial Expression Recognition and Audience Measurement - Third International Workshop, VAAM 2016, and Second International Workshop, FFER 2016, Cancun, Mexico, December 4, 2016, Revised Selected Papers*, volume 10165 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2016.

[69] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[70] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[71] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[72] Hong Liu, Liang Hu, and Liqian Ma. Online RGB-D person re-identification based on metric model update. *CAAI Trans. Intell. Technol.*, 2(1):48–55, 2017.

[73] Kai Liu, Zheng Xu, Zhaohui Hou, Zhicheng Zhao, and Fei Su. Further non-local and channel attention networks for vehicle re-identification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2494–2500. Computer Vision Foundation / IEEE, 2020.

[74] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[75] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *CoRR*, abs/2205.13542, 2022.

[76] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2016.

[77] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[78] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[79] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[80] Christian Lusardi, Abu Md Niamul Taufique, and Andreas E. Savakis. Robust multi-object tracking using re-identification features and graph convolutional networks. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*, pages 3861–3870. IEEE, 2021.

[81] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[82] Andrii Maksai, Xinchao Wang, François Fleuret, and Pascal Fua. Non-markovian globally consistent multi-object tracking. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2563–2573. IEEE Computer Society, 2017.

[83] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11164–11173. Computer Vision Foundation / IEEE, 2020.

[84] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 8834–8844. IEEE, 2022.

[85] Anton Milan, Seyed Hamid Rezatofighi, Anthony R. Dick, Ian D. Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4225–4232. AAAI Press, 2017.

[86] Raha Moraffah, Mansooreh Karami, Ruocheng Guo, Adrienne Raglin, and Huan Liu. Causal interpretability for machine learning - problems, methods and evaluation. *CoRR*, abs/2003.03934, 2020.

[87] Matteo Munaro, Andrea Fossati, Alberto Basso, Emanuele Menegatti, and Luc Van Gool. One-shot person re-identification with a consumer depth camera. In Shaogang Gong, Marco Cristani, Shuicheng Yan, and Chen Change Loy, editors, *Person Re-Identification*, Advances in Computer Vision and Pattern Recognition, pages 161–181. Springer, 2014.

[88] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 14200–14213, 2021.

[89] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14200–14213. Curran Associates, Inc., 2021.

[90] Tanmayee Narendra, Anush Sankaran, Deepak Vijaykeerthy, and Senthil Mani. Explaining deep learning models using causal inference, 2018.

[91] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Trans. Autom. Control.*, 54(3):481–497, 2009.

[92] Anshul Paigwar, David Sierra-Gonzalez, Özgür Erkent, and Christian Laugier. Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2926–2933, October 2021.

[93] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learning for multiple object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 164–173. Computer Vision Foundation / IEEE, 2021.

[94] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection, 2020.

[95] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10386–10393, 2020.

[96] Su Pang, Daniel Morris, and Hayder Radha. Fast-clocs: Fast camera-lidar object candidates fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 187–196, January 2022.

[97] Ziqi Pang, Zhichao Li, and Naiyan Wang. Model-free vehicle tracking and state estimation in point cloud sequences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 8075–8082. IEEE, 2021.

[98] Ziqi Pang, Zhichao Li, and Naiyan Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino, editors, *Computer Vision - ECCV 2022 Workshops - Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part I*, volume 13801 of *Lecture Notes in Computer Science*, pages 680–696. Springer, 2022.

[99] Cosimo Patruno, Roberto Marani, Grazia Cicirelli, Ettore Stella, and Tiziana D'Orazio. People re-identification using skeleton standard posture and color descriptors from RGB-D data. *Pattern Recognit.*, 89:77–90, 2019.

[100] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[101] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85. IEEE Computer Society, 2017.

[102] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5099–5108, 2017.

[103] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2B: point-to-box network for 3d object tracking in point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 6328–6337. Computer Vision Foundation / IEEE, 2020.

[104] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *CoRR*, abs/2206.04670, 2022.

[105] Wen Qian, Hao Luo, Silong Peng, Fan Wang, Chen Chen, and Hao Li. Unstructured feature decoupling for vehicle re-identification. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XIV*, volume 13674 of *Lecture Notes in Computer Science*, pages 336–353. Springer, 2022.

[106] Luc de Raedt, Sebastijan Dumančić, Robin Manhaeve, and Giuseppe Marra. From statistical relational to neuro-symbolic artificial intelligence. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial*

Intelligence, IJCAI-20, pages 4943–4950. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Survey track.

[107] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[108] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[109] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[110] Tara Sadjadpour, Jie Li, Rares Ambrus, and Jeannette Bohg. Shasta: Modeling shape and 0031spatio-temporal affinities for 3d multi-object tracking. *CoRR*, abs/2211.03919, 2022.

[111] Rick Salay, Krzysztof Czarnecki, Hiroshi Kuwajima, Hirotoshi Yasuoka, Toshihiro Nakae, Vahdat Abdelzad, Chengjie Huang, Maximilian Kahn, and Van Duong Nguyen. The missing link: Developing a safety case for perception components in automated driving, 2022.

[112] Ricardo Sanchez-Matilla, Fabio Poiesi, and Andrea Cavallaro. Online multi-target tracking with strong and weak detections. In Gang Hua and Hervé Jégou, editors, *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, volume 9914 of *Lecture Notes in Computer Science*, pages 84–99, 2016.

[113] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[114] Clint Sebastian, Raffaele Imbriaco, Egor Bondarev, and Peter H. N. de With. Dual embedding expansion for vehicle re-identification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2475–2484. Computer Vision Foundation / IEEE, 2020.

[115] Shai Shalev-Shwartz and Amnon Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training, 2016.

[116] Cosma Rohilla Shalizi. *Advanced Data Analysis from an Elementary Point of View*. 2021.

[117] Jiayao Shan, Sifan Zhou, Yubo Cui, and Zheng Fang. Real-time 3d single object tracking with transformer. *CoRR*, abs/2209.00860, 2022.

[118] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[119] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[120] Guang Shu, Afshin Dehghan, Omar Oreifej, Emily M. Hand, and Mubarak Shah. Part-based multiple-person tracking with partial occlusion handling. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 1815–1821. IEEE Computer Society, 2012.

[121] Vishwanath A. Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. *CoRR*, abs/1904.01649, 2019.

[122] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[123] Kaylene Caswell Stocking, Alison Gopnik, and Claire Tomlin. From robot learning to robot understanding: Leveraging causal graphical models for robotics. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1776–1781. PMLR, 08–11 Nov 2022.

[124] Autonomous Stuff. Alpha prime, powering safe autonomy.

[125] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud

processing. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2530–2539. Computer Vision Foundation / IEEE Computer Society, 2018.

[126] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 2443–2451. Computer Vision Foundation / IEEE, 2020.

[127] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *CoRR*, abs/2012.15460, 2020.

[128] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[129] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 2021.

[130] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[131] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[132] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11794–11803, June 2021.

[133] Li Wang, Xinyu Zhang, Wenyuan Qin, Xiaoyu Li, Lei Yang, Zhiwei Li, Lei Zhu, Hong Wang, Jun Li, and Huaping Liu. CAMO-MOT: combined appearance-motion optimization for 3d multi-object tracking with camera-lidar fusion. *CoRR*, abs/2209.02540, 2022.

[134] Tianyu Wang, Xiaowei Hu, Zhengzhe Liu, and Chi-Wing Fu. Sparse2dense: Learning to densify 3d features for 3d object detection. *CoRR*, abs/2211.13067, 2022.

[135] Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multi-modal classification networks hard? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[136] Yikai Wang, Wenbing Huang, Fuchun Sun, Tingyang Xu, Yu Rong, and Junzhou Huang. Deep multimodal fusion by channel exchanging. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[137] Yikai Wang, Wenbing Huang, Fuchun Sun, Tingyang Xu, Yu Rong, and Junzhou Huang. Deep multimodal fusion by channel exchanging. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4835–4845. Curran Associates, Inc., 2020.

[138] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):146:1–146:12, 2019.

[139] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XI*, volume 12356 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 2020.

[140] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *IEEE/RSJ International Conference on*

Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021, pages 10359–10366. IEEE, 2020.

[141] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M. Kitani. GNN3DMOT: graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 6498–6507. Computer Vision Foundation / IEEE, 2020.

[142] John Willes, Cody Reading, and Steven L. Waslander. Intertrack: Interaction transformer for 3d multi-object tracking. *CoRR*, abs/2208.08041, 2022.

[143] Nan Wu, Stanisław Jastrzebski, Kyunghyun Cho, and Krzysztof J. Geras. Characterizing and overcoming the greedy nature of learning in multi-modal deep neural networks, 2022.

[144] Xiaopei Wu, Liang Peng, Honghui Yang, Liang Xie, Chenxi Huang, Chengqi Deng, Haifeng Liu, and Deng Cai. Sparse fuse dense: Towards high quality 3d detection with depth completion. *arXiv preprint arXiv:2203.09780*, 2022.

[145] Kevin Muyuan Xia, Kai-Zhan Lee, Yoshua Bengio, and Elias Bareinboim. The causal-neural connection: Expressiveness, learnability, and inference. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[146] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4705–4713. IEEE Computer Society, 2015.

[147] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-temporal relation networks for multi-object tracking. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3987–3997. IEEE, 2019.

[148] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Zhou Bin, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3047–3054, 2021.

[149] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pages 90–105. Springer, 2018.

[150] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018.

[151] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[152] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Deep metric learning for person re-identification. In *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*, pages 34–39. IEEE Computer Society, 2014.

[153] Kexin Yi*, Chuang Gan*, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020.

[154] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[155] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 11784–11793. Computer Vision Foundation / IEEE, 2021.

[156] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Multimodal virtual point 3d detection. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16494–16507. Curran Associates, Inc., 2021.

[157] Jin Hyeok Yoo, Yecheol Kim, Jisong Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object

detection. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 720–736, Cham, 2020. Springer International Publishing.

[158] Ju Hong Yoon, Chang-Ryeol Lee, Ming-Hsuan Yang, and Kuk-Jin Yoon. Online multi-object tracking via structural constraint event aggregation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1392–1400. IEEE Computer Society, 2016.

[159] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[160] Jan-Nico Zaech, Alexander Liniger, Dengxin Dai, Martin Danelljan, and Luc Van Gool. Learnable online graph representations for 3d multi-object tracking. *IEEE Robotics Autom. Lett.*, 7(2):5103–5110, 2022.

[161] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. MOTR: end-to-end multiple-object tracking with transformer. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXVII*, volume 13687 of *Lecture Notes in Computer Science*, pages 659–675. Springer, 2022.

[162] Kaiwei Zeng, Munan Ning, Yaohua Wang, and Yang Guo. Hierarchical clustering with hard-batch triplet loss for person re-identification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13654–13662. Computer Vision Foundation / IEEE, 2020.

[163] Yihan Zeng, Chao Ma, Ming Zhu, Zhiming Fan, and Xiaokang Yang. Cross-modal 3d object detection and tracking for auto-driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 3850–3857. IEEE, 2021.

[164] Haolin Zhang, Dongfang Yang, Ekim Yurtsever, Keith A. Redmill, and Ümit Özgüner. Faraway-frustum: Dealing with lidar sparsity for 3d object detection using fusion. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2646–2652, 2021.

[165] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *Int. J. Comput. Vis.*, 129(11):3069–3087, 2021.

[166] Cairong Zhao, Xinbi Lv, Zhang Zhang, Wangmeng Zuo, Jun Wu, and Duoqian Miao. Deep fusion feature representation learning with hard mining center-triplet loss for person re-identification. *IEEE Trans. Multim.*, 22(12):3180–3195, 2020.

[167] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 16239–16248. IEEE, 2021.

[168] Jianan Zhao, Fengliang Qi, Guangyu Ren, and Lin Xu. Phd learning: Learning with pompeiu-hausdorff distances for video-based vehicle re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 2225–2235. Computer Vision Foundation / IEEE, 2021.

[169] Meng Zheng, Srikrishna Karanam, Ziyan Wu, and Richard J. Radke. Re-identification with consistent attentive siamese networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 5735–5744. Computer Vision Foundation / IEEE, 2019.

[170] Zhedong Zheng, Minyue Jiang, Zhigang Wang, Jian Wang, Zechen Bai, Xuanmeng Zhang, Xin Yu, Xiao Tan, Yi Yang, Shilei Wen, and Errui Ding. Going beyond real data: A robust visual representation for vehicle re-identification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 2550–2558. Computer Vision Foundation / IEEE, 2020.

[171] Zhedong Zheng, Xiaohan Wang, Nenggan Zheng, and Yi Yang. Parameter-efficient person re-identification in the 3d space. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2022.

[172] Qinqin Zhou, Bineng Zhong, Xiangyuan Lan, Gan Sun, Yulun Zhang, Baochang Zhang, and Rongrong Ji. Fine-grained spatial alignment model for person re-identification with focal triplet loss. *IEEE Trans. Image Process.*, 29:7578–7589, 2020.

[173] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[174] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection, 2019.

[175] Xiangyu Zhu, Zhenbo Luo, Pei Fu, and Xiang Ji. Voc-reid: Vehicle re-identification based on vehicle-orientation-camera. *CoRR*, abs/2004.09164, 2020.

Appendix A

Extended experimental details

A.1 Trackers

Method Name	Modality	AMOTA↑	AMOTP↓	MOTAR ↑	MOTA↑	MOTP↓	RECALL↑	GT	MT↑	ML↓	FAF↓	TP↑	FP↓	FN↓	IDS↓	FRAG↓	TID↓	LGD↓
CAMO-MOT[133]	CL	0.763	0.527	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AlphaTrack[163]	CL	0.732	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
*SimpleTrack(2 Hz)[98]	CL	0.731	0.598	0.837	0.645	0.330	0.776	14556	4772	1124	45.3	85220	13241	16106	571	549	0.48	0.79
Intertrack[163]	CL	0.721	0.566	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
+Differentiable Tracker (ours)	CL	0.718	0.657	0.827	0.636	0.340	0.769	14556	4534	1246	44.8	83779	12767	17379	739	546	0.64	0.94
Eager MOT w/o images [58]	CL	0.712	0.569	–	–	–	0.752	–	–	–	–	–	–	–	899	–	–	–
NEBP[67]	L	0.708	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ShaSTA[110]	L	0.703	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
BP[67]	L	0.698	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SimpleTrack(10hz)[98]	L	0.696	0.547	–	0.602	–	–	–	–	–	–	–	–	–	405	–	–	–
OGR3MOT[160]	L	0.693	0.627	–	0.602	–	–	–	–	–	–	–	–	–	262	332	–	–
SimpleTrack(2 Hz)[98]	L	0.687	0.573	–	0.592	–	–	–	–	–	–	–	–	–	519	–	–	–
CenterPoint[155]	L	0.665	0.567	–	0.562	–	–	–	–	–	–	–	–	–	562	424	–	–
*Probabilistic MOT [15]	CL	0.658	0.672	0.834	0.582	0.342	0.699	14556	3643	1793	44.1	75488	12858	25706	703	422	0.93	1.24
Eager MOT w/o images [58]	L	0.651	0.587	–	–	–	0.698	–	–	–	–	–	–	–	864	–	–	–

*: Results that we replicated using BEVfusion C+L detections; +:Models we implemented

Table A.1: **Tracking results reported on the nuScenes validation set for various existing methods.** Despite being devoid of re-identification information, the trackers we use to evaluate our ReID methods have strong performance compared to SOTA. This demonstrates they are relevant baselines from which to extract tracking errors. We note that CAMO-MOT, the first-place entry, leverages an image ReID model as part of their pipeline.

We collected tracking mistakes from three different 3D multi-object trackers to evaluate the performance of our point cloud re-identification networks. We chose two publicly available baselines: [15, 98] and implemented a third tracker ourselves. We used the detections of BEVfusion C+L (table 4.2) as input to each tracker.

The entry designated "*Probabilistic MOT" in table A.1 uses default open source code from[1]. This tracker achieves the lowest AMOTA of the three methods we ran ourselves and we find that its errors are easier for a point cloud ReID network to classify correctly than errors from a more performant tracker (see figure 5.6).

The entry designated "*SimpleTrack(2 Hz)" in the table uses existing open-source code.[2] This tracker performs the strongest of the three that we evaluated ourselves. We use the 2 Hz variant of the simple track algorithm, keeping all hyperparameters set to their default values.

The entry designated "+Differentiable Tracker (ours)" in table A.1 is the tracker we implemented ourselves. We chose to implement a differentiable tracker (one that learns scores for Hungarian matching) as our study would be incomplete without one. Many differentiable trackers are advertised in the literature, but none release their open source code at the time of writing [110, 142, 66, 133]. Our tracking pipeline follows the general structure of tracking-by-detection, where bounding box features $(x, y, z, \sin(\theta), \cos(\theta), v_x, v_y)$ along with the ego's velocity (in the xy plane) and yaw rate are input to an MLP similar to [66]. We chose not to use any bounding box size features so as to remove any potential re-identification information from the tracker itself. The MLP's representation is fed to an LSTM to encode track representations. We use track refinement and prediction modules similar to [66]. To match tracks to detections, we first model associations between objects using cross-attention as done in [142]. Next, we concatenate track and detection representations into pairs, feeding them to an MLP to regress matching scores. Individual representations of tracks and detections are also fed to MLPs to regress scores for being a false positive detection or newborn detection and false positive track or false positive detection, respectively. This is similar to what is done in [110]. To train our model, we emulate the inference phase during training as is done in [66]. Using teacher forcing, we train our model with losses similar to [110] for 20 epochs on the nuScenes dataset. Each epoch contains one sequence of 16 frames sampled for each scene in the training set. Gradients are accumulated over 8 sequences (128 frames) before taking a step. We used the AdamW [77] optimizer with a learning rate of $1e - 4$, weight decay of 0.01, and cosine learning rate and momentum schedules [122]. We note that the object detector is frozen throughout training.

[1] We used the official implementation of probabilistic multi-object tracking found here `https://github.com/eddyhkchiu/mahalanobis_3d_multi_object_tracking`.

[2] We used the official implementation of simple track found here `https://github.com/tusen-ai/SimpleTrack`.

A.2 Hyperparameters

Table. A.2 characterizes the hyperparameters of our ReID networks in greater detail. We note that the difference in training epochs (400 vs. 500) for Waymo and nuScenes datasets was intentional to have approximately the same number of gradient descent steps for each (±3 epochs).

Parameter	Explanation	Value
Shared hyperparameters		
WD	weight decay	0.01
Point hyperparameters		
D_p	Point input dimension	3
N	Number of input points	128
LR	Learning Rate	$3e-4$
B	Batch Size	256×4
$S \times d_L$	Sequence length and latent feature dimension	128×64
Image hyperparameters		
D_I	Image input dimension	$3 \times 224 \times 224$
B	Batch Size	64×4
LR	Learning Rate	$1e-5$
Randomly initialized Waymo models		
E	Training epochs	400
Randomly initialized nuScenes models		
E	Training epochs	500
Pretrained Models		
E	Training epochs	200

Table A.2: **Hyperparameters of our different architectures.**

A.3 Additional results analysis.

While the main manuscript highlights the results that are central to our contributions, some complementary figures could not be included. We include them and a corresponding discussion here.

A.3.1 Why does the sampling algorithm have a positive bias for objects at higher point densities?

Figure A.1 illustrates a limitation of the sampling algorithm we use to select positive and negative samples during training. While sampling negative examples uniformly at random is an intuitive choice, it actually introduces a positive bias at higher point densities. This is because relatively few objects actually contain samples that are of high density, therefore, the probability of sampling one from all possible observations is low. In contrast, when sampling positive examples our sampling algorithm must by definition sample other observations of the same object. Therefore, if the first observation has many points, then it is more likely that other observations of that same object will also have many points. This is illustrated in the bottom plot of figure A.1, where we graph the *ratio* of the probability of sampling a positive pair of observations with both observations containing at least x points *and* the probability of sampling a negative with the same characteristics. As such, the y-axis quantifies the extent to which the sampling is imbalanced in favor of positive examples. We observe that the pedestrian class experiences the greatest imbalance of all classes, where the ratio of positive to negative pairs is as high as 1000 to 1 for 2048+ points. Correspondingly, we observe (in the top left plot) that the accuracy of our point ReID networks decreases as the number of points is increased for the pedestrian class. This would go unnoticed if the same sampling algorithm were used at test time. However, we opted to use a balanced sampling algorithm at testing time to expose any potential biases learned during training.

By also plotting the positive and negative F1-scores, we can gain further insight into the drop in accuracy and confirm our hypothesis that the networks learn the bias that is present in the training data. For both pedestrian and car, we see the same trend: the positive F1-score is higher than the negative F1-score—confirming the positive bias. One might still ask: but why does the accuracy for the car class seem unaffected, while the pedestrian accuracy suffers greatly? This is because of the magnitude of the imbalance. Note that the y-axis of the bottom figure is log scale. We note that this imbalance phenomenon is also observed for images, despite their input being devoid of explicit information about the point density. We hypothesize that the image models are able to find correlations between the input images and point density which can then be used to fit the imbalance in the training distribution. When inspecting cropped images with a large number of LiDAR points, we notice that the length of most images is compressed to fit the input size of 224×224— a correlation that an image model should be able to pick up on.

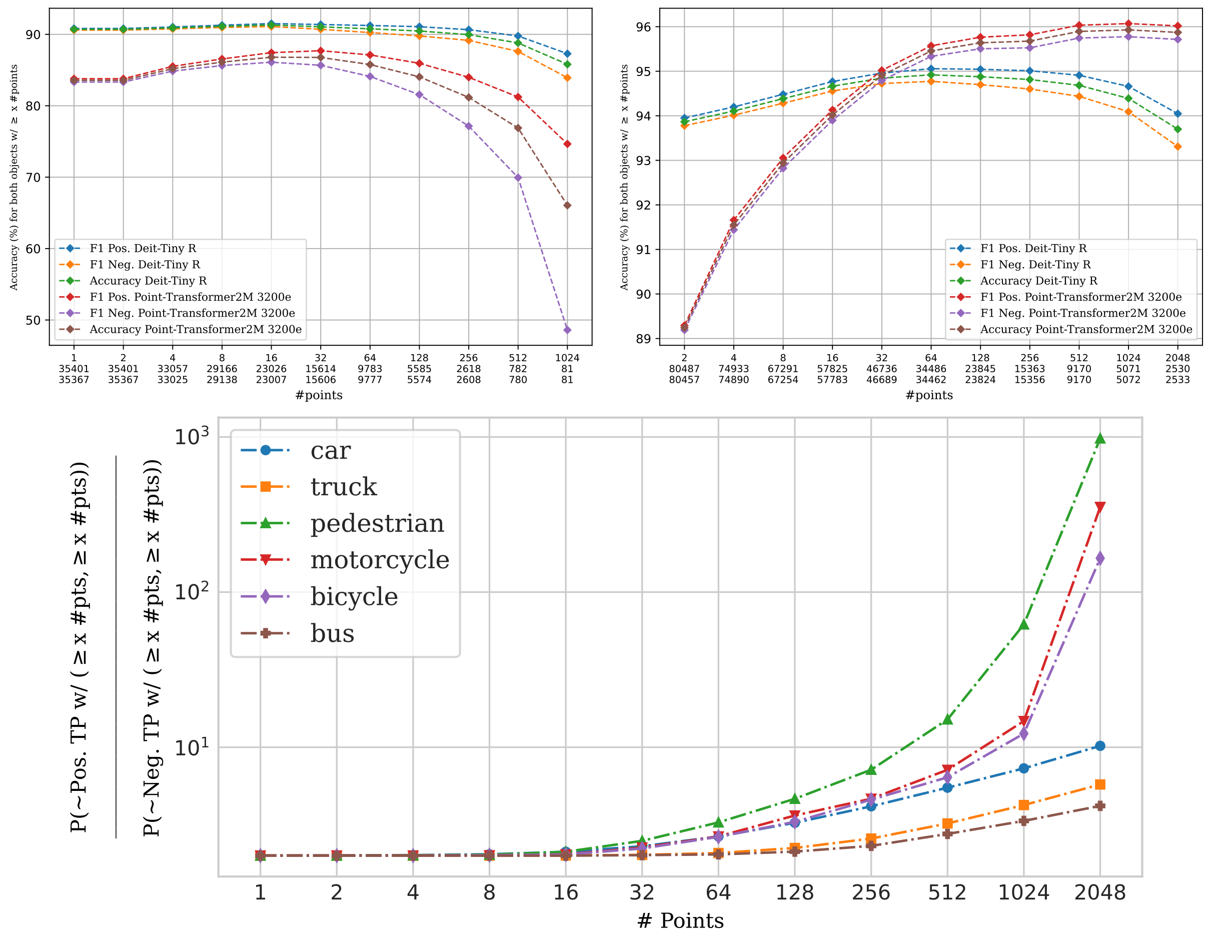

Figure A.1: **Unbalanced train-time sampling algorithm (bottom) causes our point ReID models to be positively biased (top left) at high point densities.** The y-axis of the bottom plot quantifies the extent to which the sampling is imbalanced in favor of positive examples as the number of points is increased. The top two plots, pedestrian (left) and car (right), show the effect of the sampling algorithm on network performance. We note that it dramatically affects the performance of our strongest-performing network on the pedestrian class.

A.3.2 Additional analysis comparing accuracy at different point densities.

Figure A.2 is the twin of figure 5.2 from the main manuscript. Each figure plots the performance of our ReID networks on nuScenes (left) and WOD (right) at different point densities. The difference between the two figures is seen on the x-axis. Figure A.2 plots accuracy on pairs of observations with at least one observation containing x points or more, while figure 5.2 requires that both observations in the pair have at least x points. We observe that the increase in accuracy is steeper when restricting to cases where both observations have more than x points. However, figure 5.2 does show the same trend but with a smaller slope.

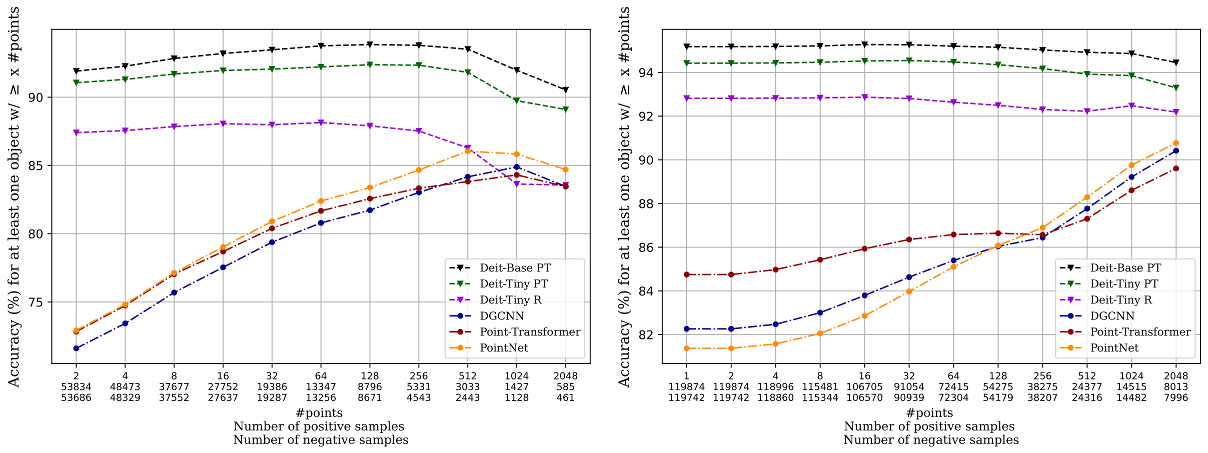

Figure A.2: **The performance of point cloud ReID approaches image ReID with sufficient points.** The plot shows the performance of the image and point cloud ReID networks as a function of the number of points in at least one observation of the pair. Left was trained on nuScenes and evaluated on the *nuScenes eval* set, while right was trained on Waymo and evaluated on the *Waymo eval 1*.

A.4 Dataset visualization

Figures A.3 and A.4 provide visual examples of samples from our WOD ReID dataset. The images are cropped from projected 3D bounding boxes predicted by our CenterPoint model (see sec. 4 for details). The center plots shows the LiDAR crop associated with the same bounding box. The rightmost plot shows complete point clouds created by aggregating the points from all observations using ground truth bounding boxes. Aggregated deformable objects (pedestrian, bicycle, and motorcycle) have a blob-like appearance, while rigid objects retain a more detailed shape. This is due to their deformability, causing them to take on many different poses over a sequence. We selected observations with more than 200 points so that it is possible for a human to make out the underlying object. We note, however, that most of the observations on the dataset contain fewer than 200 points.

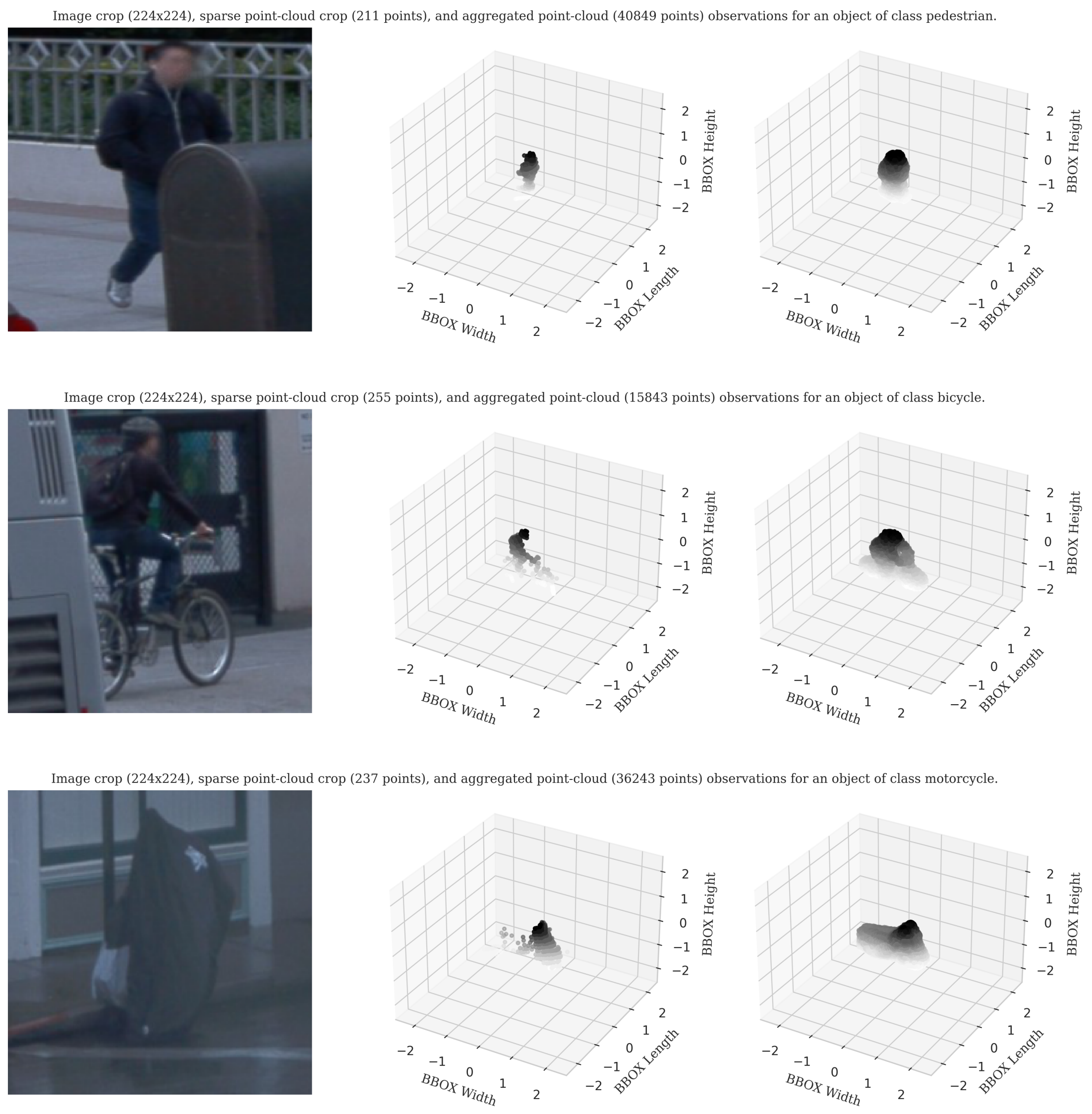

Figure A.3: **Samples from our Waymo ReID dataset for deformable objects.** The plot shows the cropped image (left) and cropped sparse point cloud (center) for the same predicted 3D bounding box (output by our CenterPoint model). We also include the corresponding complete version of the point cloud (right), used to train our model in the ablation study (see table. 5.4).

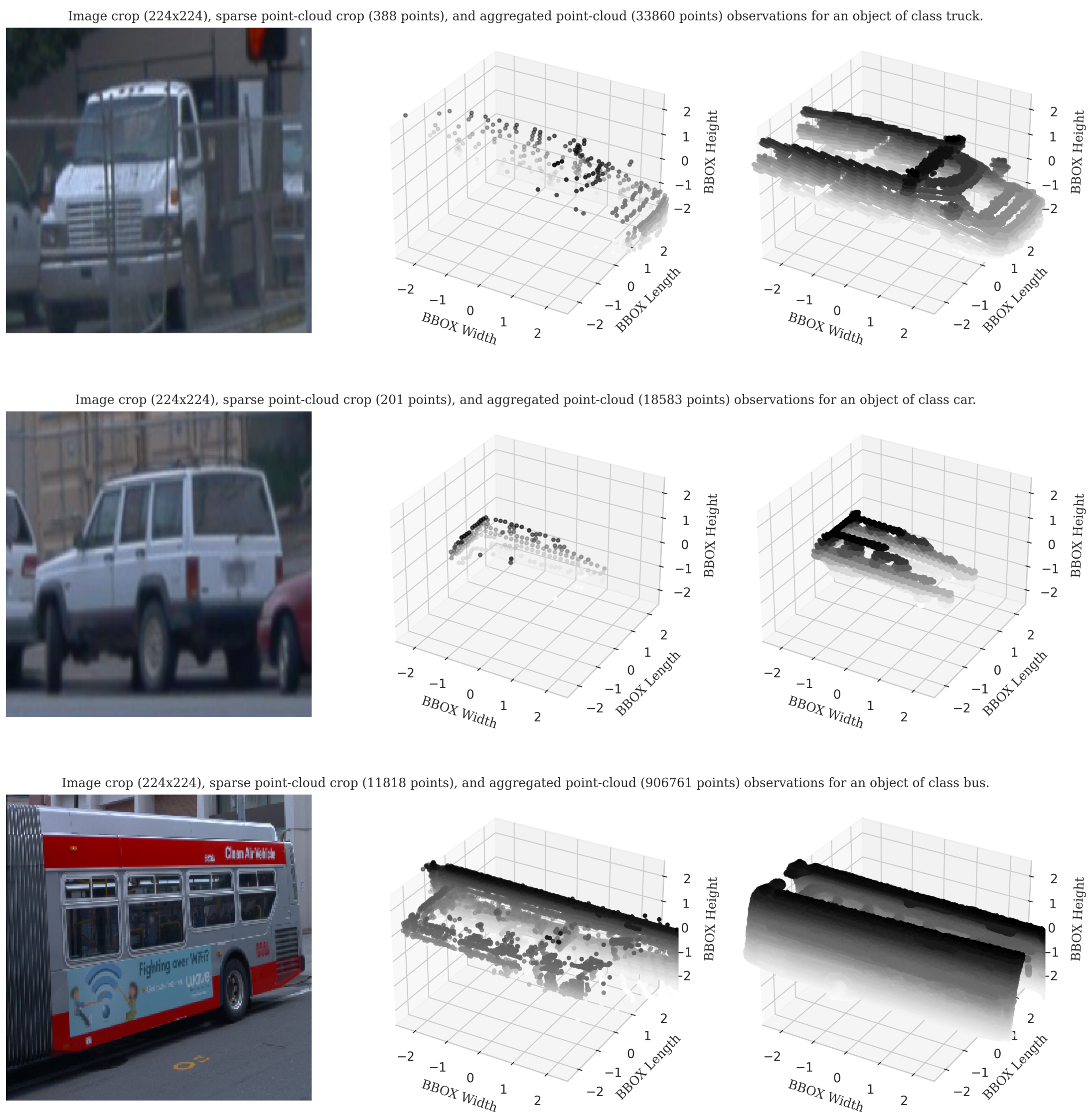

Figure A.4: **Samples from our Waymo ReID dataset continued for rigid objects.** The plot shows the cropped image (left) and cropped sparse point cloud (center) for the same predicted 3D bounding box (output by our CenterPoint model). We also include the corresponding complete version of the point cloud (right), used to train our model in the ablation study (see table. 5.4).