CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Multimodal Data Fusion for BEV Perception

Masters thesis in Master's Programme in Data Science and AI

YUNER XUAN

YING QU

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Multimodal Data Fusion for BEV Perception

YUNER XUAN
YING QU

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Multimodal Data Fusion for BEV Perception
YUNER XUAN
YING QU

Supervisor: Selpi, Department of Computer Science and Engineering
Examiner: Marina Axelson-Fisk, Department of Mathematical Sciences

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2024

Multimodal Data Fusion for BEV Perception
YUNER XUAN
YING QU
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In autonomous driving, sensors are situated across different parts of the vehicle to capture the information from the surrounding environments to allow the autonomous vehicles to address various tasks related to driving decisions, like object detection, semantic segmentation and path planning. In the diverse approaches of perception, birds-eye-view (BEV) perception has progressed impressively over recent years. In contrast to front-view or perspective view modalities, BEV provides a comprehensive representation of the vehicles surrounding environment, which is fusion-friendly and offering convenience for downstream applications.

As vehicle cameras are oriented outward and parallel to the ground, the captured images are in a perspective view that is perpendicular to the BEV. Consequently, a crucial part of BEV perception is the transformation of multi-sensor data from perspective view (PV) to BEV. The quality and efficiency of this transformation play a critical role in influencing the performance of subsequent specific tasks.

This thesis project aims to study comprehensive multimodal data fusion solutions for PV-to-BEV transformation. We analyzed the common and unique characteristics of existing approaches and assessed their performance against a selected downstream perception task, focusing on object detection within a short distance. Additionally, we implemented mainly two modules Global Position Encoding (GPE) and Information Enhanced Decoder (IED) to enhance the performance of the multi-modal data fusion model.

Keywords: Multi Modality, Sensor Fusion, BEV Perception, LiDAR, Camera, Transformer, Deep learning, 3D Object Detection, thesis.

Acknowledgements

We would like to express our thanks to Selpi at Chalmers University of Technology for providing us the opportunity of exploring multimodal data fusion methods in this Master's thesis. We would also like to thank Marina Axelson-Fisk at Chalmers University of Technology for being our examiner.

Yuner Xuan and Ying Qu, Gothenburg, 2024-05-20

Contents

List of Figures

List of Tables

1

Introduction

In the realm of autonomous driving, an array of sensors situated across different parts of the vehicle to capture the information from the surrounding environments. This understanding of the environment empowers the autonomous vehicles to address various tasks related to driving decisions, including object detection, semantic segmentation and path planning.

Among diverse approaches of perception in autonomous driving, birds-eye-view (BEV) perception has progressed impressively over recent years. In contrast to front-view or perspective view modalities, BEV provides a comprehensive representation of the vehicles surrounding environment, which is fusion-friendly and offering convenience for downstream applications.

As vehicle cameras are oriented outward and parallel to the ground, the captured images are in a perspective view that is perpendicular to the BEV. Consequently, a crucial part of BEV perception is the transformation of multi-sensor data from perspective view (PV) to BEV. The quality and efficiency of this transformation play a critical role in influencing the performance of subsequent specific tasks. However, the fusion of multimodal data and the generation of BEV features remain challenging.

This thesis project aims to study comprehensive multimodal data fusion solutions for PV-to-BEV transformation. We will analyze the common and unique characteristics of existing approaches and assess their performance against a selected downstream perception task, focusing on object detection within a short distance. Additionally, we aspire to contribute to this field by designing a novel method and evaluating its performance.

1.1 Background

1.1.1 PV-BEV Transformation

The PV-BEV transformation is an algorithm designed to convert perspective-view inputs, such as images captured by a camera and point clouds provided by LiDAR, into BEV features. This transformation facilitates the fusion of features from different modalities under a unified representation. Various approaches have emerged to address the PV-BEV transformation, which can be classified into depth-based,

MLP-based, and transformer-based methods. Depth-based transformation relies on geometry, while MLP-based and transformer-based methods leverage the power of deep neural networks to convert features in the input data into BEV features [1].

1.1.2 Multimodal Data Fusion for Camera and LiDAR data

Different sensors equipped in autonomous vehicles provide diverse information about the vehicle's surroundings. Camera data contains dense color and texture information, LiDAR provides accurate depth and structural details. The fusion of data from different sensors complements the overall information and enhances the performance of downstream tasks, such as object detection[2].

However, when fusing these features, the challenge of misalignment or inaccurate depth prediction between camera and LiDAR data needs to be addressed. Careful consideration and handling are required to align and integrate features at different stages of the fusion process.

1.2 Project Scope

This project is dedicated to exploring the domain of Modality Feature Generation and BEV (Birds-Eye-View) Feature Encoder, with a primary goal of comparing the performance of various fusion methods in transforming PV (Perspective View) to BEV.

The experimental framework of this project will leverage the NuScenes dataset[3] , employing three baseline models inspired by notable works: BEVFusion[4], DeepInteraction[5], and CMT[6]. In addition to analyzing these established methods, a novel method tailored for multimodal data fusion is designed and developed for Bird's Eye View (BEV) perception.

1.3 Thesis Structure

The structure of the thesis is as follows:

Chapter 2 provides the problem definition of multi-modality data fusion, covering the theoretical background, concepts, and an in-depth study of three foundational methods essential for understanding this thesis.

Chapter 3 describes the specific methods and approaches employed in this thesis, detailing the design and implementation strategies.

Chapter 4 includes information about the data utilized, evaluation metrics, and the experimental setup designed to test the approach.

Chapter 5 presents an analysis of the results obtained from the experiments described in Chapter 4, offering insights into the performance and implications of the findings.

Chapter 6 discusses the limitations of this thesis, potential ideas for future research, and considerations of the risks and ethical aspects related to the study.

Chapter 7 summarizes the results and discussions, highlighting the main contributions and conclusions drawn from this thesis.

2

Theory

2.1 Problem Definition

To define the multi-modality data fusion problem formally, we divide the whole system into three parts.

a. Modality Features Generation

The system will firstly process the raw data from camera and point clouds into modality features $\mathbf{h}_c, \mathbf{h}_p$ respectively.

Modality features can be considered as the results from some existing feature extraction models. We can choose image feature generation model from ResNet[7], Dual-Swin[8] and DLA[9] backbone networks. PointPillars[10], CenterPoint[11] and TransFusion[12] could be candidates for the LiDAR feature generation.

b. BEV Feature Encoder

Once $\mathbf{h}_c, \mathbf{h}_p$ are generated, we can use a certain fusion method to generate BEV feature:

$$\mathbf{h}_{BEV} = Fusion\_Encoder(\mathbf{h}_c, \mathbf{h}_p)$$

c. Prediction Decoder

Following the BEV feature encoder, the decoder module gives Region of Interest (RoI) features $\{\mathbf{R}_n\}_{n=1}^{N}$, where N denotes the number of candidates of detected objects:

$$\{\mathbf{R}_n\}_{n=1}^{N} = Prediction\_Decoder(\mathbf{h}_{BEV})$$

The RoI representation $\{\mathbf{R}_n\}_{n=1}^{N}$ is further used to generate object detection results including position, dimension, orientation and velocity as required for evaluation.

2.2 Previous Work

2.2.1 Fusion Paradigms

The common fusion paradigms mainly includes: early-fusion(data-level), middle-fusion(feature-level) and late-fusion(results-level)[13]–[15]. Recent studies on BEV data-fusion for autonomous driving have made impressive progress. Most of them [4], [12], [13] fall in the middle-fusion or feature-level category. The BEV data-fusion based on middle fusion paradigm have been implemented in various architectures to efficiently extract features both from LiDAR and camera data.

2.2.2 Data-level Enhancement

Katare et al.[16] introduced a method for data-level enhancement on the NuScenes dataset[3]. This method employs resampling and data augmentation techniques to balance the class distribution. As a result, it achieves improved performance on minority classes, such as cyclists and motorcyclists.

2.2.3 Pure Geometry Projection

The early studies use point clouds directly to locate and fuse corresponding 2D features by geometry projection. According to the extrinsic and intrinsic matrix of cameras, **PointPainting**[17] projects point clouds data to the position of images and extract the image features at the corresponding locations, like painting the points. Once the LiDAR data is fused with image features, it will give prediction through a 3D detection network.

2.2.4 Model-based Query Generation

Instead of only relying on the raw geometry relation, some following models process the point clouds data first. It improves the efficiency to associate multimodal data. **AutoAlignV2** [18] employs a Cross-Attention Feature Alignment module to archive the multimodal feature fusion. It also applies the data augmentation method, GT-AUG, to generate new samples for both 2D and 3D data. **FUTR3D**[19] first uses separate encoders for different modality features and employs a modality-agnostic feature sampler to obtain features through queries. The query location will be updated iteratively. **Transfusion**[12] employs a two-stage strategy. First, it generates location queries from point cloud features. Image features are then extracted by projecting these queries onto the image coordinates. Both LiDAR and camera features are fused by a transformer decoder layer, and the detection results are given by a LiDAR detection network. **Fully Sparse Fusion**[20] leverages instance segmentation to extract point-based instance features from camera and LiDAR data. These features are then fused through bi-modal instance queries and query alignment. This approach yields a faster inference speed, which is 2.7Œ faster than other state-of-the-art multimodal 3D detection methods by maintaining fully sparse characteristics.

2.2.5 Feature Extraction in Parallel

Later, researchers found that it is beneficial to extract features from both 2D and 3D data, as opposed to primarily extracting spatial information from point clouds. Previous approaches heavily rely on point clouds to generate queries. To overcome the negative effects caused by the sparsity of LiDAR data, **BEVFusion**[4] generates camera and LiDAR features with separate branches and projects them into a uniform BEV space. Compared to BEVFusion, **BEVFusion4d**[21] proposes a LiDAR-guided view transformer to fuse image data into BEV features with a prior on LiDAR data. This modification allows the BEV features to contain richer spatial information. **SimDistill**[22] utilizes a teacher-student model, where BEVFusion[4] serves as the teacher model and the camera branch of BEVFusion, along with simulated LiDAR data, is used as the student model. This method employs multi-modal distillation to enhance the performance of the student model on single-modal camera inputs, leveraging LiDAR knowledge distillation. It also provides flexibility in changing the teacher model.

2.2.6 Implicit Alignment

Another approach for aligning multimodal data is position encoding. This technique is able to implicitly align 2D and 3D features, as opposed to relying only on geometric relations. **CMT**[6] designs a coordinates encoding module(CEM), which adds position encoding both on image and point clouds features. Moreover, the queries are generated with position guidance extracted from point clouds and images. The encoded location information enables object queries to interact directly with multimodal features and implicitly align multimodal data.

2.2.7 Feature Interaction Enhancement

More recent works have developed various approaches to enforce interactions among multimodal features. Most of them are self-attention mechanisms developed based on transformer-like architectures. These methods augment the representation for each modality feature. **UniBEV**[23] employs shared queries to achieve data fusion across modalities. In addition, it designs channel normalization weights and performs channel-wise summation to generate BEV features. Instead of generating cross-modal features, **Deepinteraction**[5] maintains point clouds and image representations both for encoder and decoder modules. Attention-based feature interaction is used in the encoder. Its decoder applies multi-modal predictive interaction layers to utilize point LiDAR or camera representations to refine bounding-box prediction and let them interact mutually. **Unitr**[24] includes both mode-specific and mode-agnostic partitions. Dynamic set partitioning performs attention by sharing self-attention weights. Its modality-agnostic partition proposes a transformer backbone and shares parameters for both image and point cloud data. By sharing weights, it reduces the latency and also achieves a significant improvement. **MV-Fusion**[25] uses image data to complement missing information from radar points, including vertical and sparse information. This is achieved by performing sensor fusion with an image-guided radar transformer and a radar-guided fusion transformer.

Sparse Dense Fusion[26] performs fusion using a sparse fusion module on the voxels to form the BEV features, and a dense fusion module via cross-attention on the transformer and temporal information.

2.2.8 Task-aware Feature Extraction

Most current methods generate BEV features first and pass them to downstream tasks, which may incur information loss. A possible remedy is to extract task-related features at an earlier stage. To fully utilize the depth information from the LiDAR sensor, it directly queries the voxel and image features before BEV feature generation. **FusionFormer**[27] directly queries the voxel and image features before BEV feature generation. It alleviates potential information loss during the PV to BEV transformation.

2.3 Related Techniques of BEV Object Detection

2.3.1 Coordinates Transferring

We give some explanation of coordinates transformation involved in BEV perception in this section.

From LiDAR to camera coordinates

The transformation between LiDAR and camera coordinates includes rotation and translation in 3D, which can be presented with rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$. $\mathbf{R}$ consists of three matrix for each axis, which are:

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}, \mathbf{R}_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}, \mathbf{R}_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where θ is the rotation for each axis respectively.
Generally, the transformation with homogeneous coordinate has the form as:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{pmatrix}$$

where $(X_c, Y_c, Z_c)^T$ is the coordinate of camera, and $(X_l, Y_l, Z_l)^T$ is the coordinate of LiDAR.

From camera to image coordinates

The perspective relation from camera coordinate to image coordinate can be expressed as:

$$Z_c \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} K|\mathbf{0} \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix},$$

where x, y is the position in image coordinate, f is the focal distance, and K is the intrinsic parameters matrix.
Noting that it may encounter the ill-posed problem when applying image to camera coordinations transformation without the depth information.

From image to pixel coordinates

By ignoring the lens distortion, we can represent the transformation from image to pixel coordinates as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where d_x, d_y is the physical length of a pixel on the image sensor, and u_0, v_0 is the center of image sensor in image coordination.

In summary, the transformation between pixel and LiDAR coordinates is:

$$\begin{aligned} Z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} &= \begin{pmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{pmatrix} \end{aligned}$$

where $\begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}$ is the camera intrinsic parameters and $\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$ is the extrinsic parameters between camera and LiDAR sensors.
In addition, we can transform the LiDAR coordinate into BEV directly by voxelization and project to BEV by sampling along the height axis.

2.3.2 Voxelization

BEV 3D object detection task aims to predict objects bounding box in 3D space. Each bounding box is defined as $b = (u, v, d, w, l, h, \alpha)$, (u, v, d) is the center location relative to ground plane, and (w, l, h) indicates the size of its width, length and height. α is the yaw angle, which denotes the rotation around Z-axis.

Most of recent works on BEV 3D object detection researches [4]–[6], [23] use 3D encoders to quantize points cloud data into cube bins. This process is normally named as voxelization.

There are two widely used methods to generate voxels, namely hard voxelization (HV) and dynamic voxelization (DV).

Hard Voxelization (HV)

VoxelNet [28] leverages HV method, which achieves voxelization with grouping stage and sampling stage.

Let (W, L, H) define the range of input points cloud data, and V_w, V_l, V_h define the size of each voxel, then voxelization generates $\frac{W}{V_w} \times \frac{L}{V_L} \times \frac{H}{V_h}$ voxels.

In grouping stage, one point p_i will be assigned to a spatially related voxel v_j. If the number of points are more than the pre-defined point capacity of one single voxel, it performs sub-sampling in sampling stage. The voxel sub-sampling is also required if there are more voxels than voxel capacity.

Correspondingly, once the assigned points or voxels are less than capacity, padding process is required respectively.

Dynamic Voxelization (DV)

HV mainly has two defects [29]: (1) Sub-sampling will lead to information loss and unstable training by dropping points data randomly. (2) In addition, padding process could waste computation and storage resources.

MVF [29] proposed DV, which overcomes the shortages of HV. First, the mapping relation between points and voxels are decided by flexible mapping function without mandatory sub-sampling. Moreover, the number of points and voxels is dynamic, thus padding operation is not mandatory either. The strategy of DV preserves all information and produces voxels deterministically, therefore it will generate more stable detection results.

2.3.3 Convolutional Neural Network (CNN)

A Convolutional Neural Network(CNN) [30] is a popular type of artificial neural network commonly used for pattern recognition within images and extracting image-specific features. It enables neural networks to handle tasks involving images, such as image recognition and classification.

The architecture of CNNs consists of three main types of layers: convolutional layers, pooling layers, and fully connected layers.

The convolutional layer employs kernels to scan the receptive fields of an image and recognize the presence or absence of certain features. A rectified linear unit (ReLU) layer is typically applied after the convolutional layer to introduce non-linearity to the model. The ReLU function converts all negative pixel values to zero.

The pooling layer is used to down-sample the input along its spatial dimension, reducing the number of parameters and computational complexity. Different types of pooling layers, such as max pooling and average pooling, perform downsampling differently. The maxpooling layer reduces the response maps to the maximum values. The average pooling layer reduces the response maps not to a maximum value but the average values.

The fully connected layer connects all the outputs from the previous layers to each activation unit of its layer, facilitating pattern recognition based on features extracted by the prior layer.

2.3.4 Feature Pyramid Networks(FPN)

FPN [31] is a wildly used neural network which aims to enhance the performance of detection and semantic segmentation tasks by fusing feature maps of different scales. It exploits the pyramidal architecture of features computed by convolutional networks, such that it is capable of extracting features with various scales. It is suitable for tasks involving objects of various sizes, such as object detection.

FPN consists of several modules:

1) A Backbone Network. The backbone network gradually extracts higher-level features. Typically, the scale of the feature map shrinks at higher levels.

2) A Top-down Pathway. The top-down pathway generates higher resolution feature maps from smaller scales by interpolation and upsampling.

3) Lateral connections and Fusion.

Between these two parts above, the feature maps at the neighbor level are merged by element-wise addition. The fused feature maps from each scale give predictions separately.

In our system, FPN is used in both 3D and 2D backbone networks to extract multi-scale features. In addition, FPN is also applied in detection heads to give detection results.

FPNC is a variant of FPN that extends FPN by incorporating contextual information into the feature pyramid using global average pooling. Additionally, it employs adaptive pooling and dimensionality reduction techniques to enhance computational efficiency.

CPFPN is another variation of FPN. It eliminates unused parameters and can be used with checkpointing to optimize memory usage.

2.3.5 Sparse Convolution

Sparse Convolution is widely used technique to perform convolution on sparse spatial data. It is introduce in SECOND network [29].

For 2D dense convolution, we can use $W_{u,v,l,m}$ to denote filters and $D_{u,v,l}$ to denote image elements, where u, v are spatial location indexes, l, m are input and output channels respectively. Function $P(x, y)$ gives locations with provided input location (x, y). Then the output of convolution $Y_{x,y,m}$is given as follows:

$$Y_{x,y,m} = \sum_{u,v \in P(x,y)} \sum_{l} W_{u-u_0,v-v0,l,m} D_{u,v,l},$$

where u_0, v_0 represent the kernel offset from location u, v. The author mentioned that, with a general matrix multiplication (GEMM)-based algorithm, the GEMM form computation can be given as:

$$Y_{x,y,m} = \sum_{l} W_{*,l,m} \tilde{D}_{P(x,y),l}$$

Where $\tilde{D}_{P(x,y),l}$ is generated by GEMM via gathering needed data.

Similarly, for the sparse data $D'_{i,l}$, the GEMM formulation is represented as

$$Y'_{j,m} = \sum_{l} W_{*,l,m} \tilde{D'}_{P'(j),l}$$

Where j indicates the output index, which is rearranged by removing zeros from original 2D coordinate. However, the data element $\tilde{D'}_{P'(j),l}$ generated by GEMM could still include many zeros and leads to unnecessary computations.

Instead of employing GEMM directly, Sparse Convolution constructs a computation **Rule** first, and gathers need data element without zeros. The Sparse Convolution has the form as:

$$Y'_{j,m} = \sum_{k} \sum_{l} W_{k,l,m} \tilde{D'}_{R_{k,j}k,l}$$

Where rule matrix $R_{k,j}$ provides input element index i with the given kernel offset k and output index j. Sparse Convolution can obtain the same convolution results as GEMM and avoid performing unnecessary computation on zeros.

2.3.6 Transformer

Self-attention, cross-attention and position encoding are three core modules of Transformer Network [32]. This section introduces the main idea and procedure of these modules.

Self-attention

Attention mechanism of Transformer Network involves three matrix: $Query(Q)$, $Key(K)$ and $Value(V)$, they are computed from input features with separate matrix

trainable parameters. For self-attention, Q, K and V are all generated from a same input sequence. self-attention can capture the relations between one element and the rest part of a same sequence.
The attention results can be achieved as

$$Self-attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V,$$

where d_k is the dimension of matrix, which is a scale adjustment term to remove the influence of dimension. It indicates that, the attention result can be viewed as the weighted-average of V with the weight term calculated by Q and K.
One related technique is Multi-Head Attention. Instead of using only one group of Q, K, V, one can obtain multiple groups of Q_i, K_i, V_i with independent parameters. This technique will have model features of various views.

Cross-attention

The cross attention is defined as:

$$Cross-attention(Q_s, K_t, V_t) = \text{softmax}(\frac{Q_t K_t^T}{\sqrt{d_k}})V_t$$

The major difference of Cross-attention is that, matrix Q_s is obtained from a source input sequence, while matrices K_t, V_t are computed from another target sequence. Cross-attention is used to capture the relations between one element and the elements of another sequence. For example, it is usually used in decoder module of a translation NLP model, such that one can model the mapping relations between two languages.

Position Embedding

Since the information of elements order is not include in attention computation as above, the Transformer Network[32] proposed position embedding to encode the sequence position. In that work, the authors achieved position embedding as

$$PE(position, 2i) = sin(pos/10000^{2i/d_{model}})$$
$$PE(position, 2i+1) = cos(pos/10000^{2i/d_{model}})$$

The position embedding is merged to element features by element-wise addition, such that the order information is remained.

Vision Transformer

ViT [33] firstly proposed a Transformer-based image recognition model. Unlike classic CNN, ViT divides image into a fix number of patches and passes them into a Transformer model. In our project, Transformer-based layers are employed to compute features from same modality as well as across point clouds and images. This structure is also used for refining the bounding box features in the decoder module.

2.3.7 Bounding Box Generation

Anchor-Based Method

Intuitively, one can apply 2D object detection methods to generate bounding boxes for BEV 3D object detection: the overhead map-view can be viewed as a 2D plane approximately. However, traditional 2D object detection methods generate axis-aligned bounding boxes [34], [35] , which not consider the yaw angle α. For this reason, to fit 3D detection scenario, anchor-based method not only need to produce a series of boxes with various size, but also need to consider different rotation angles. This proxy may unnecessarily have a high computation complexity with large search space. Additionally, the anchor-based method also requires anadditional process to remove unnecessary boxes based on the intersection over union rate.

Center-Based Method

CenterPoint [11] proposed an anchor-free method for 3D object detection. It mainly consists of two stages. In the first stage, it finds the center of each object by taking as input the features given by a chosen 3D points encoder model. The representation of the center point is also used to simultaneously predict the 3D box size, orientation and velocity. Specifically, the center heatmap head produces $K-$channel heatmap for each of K object class, the peak of each heatmap indicates potential a object center point. The regression head produces the size, height, and rotation angle of the box. The rotation angle is represented as sine and cosine of yaw.

In the second stage, CenterPoint gives the scores of object classes and refines the 3D box with point features of each center-point face belonging to the box region given from stage one.

In contrast to anchor-based methods, this center-based approach avoids building boxes with various predefined sizes and rotation angles. It also skips the process of deciding whether to hold or drop a candidate detection.

2.3.8 Matching Loss

In this subsection, we introduce the computation of matching loss.

The matching loss we employed in our project is firstly proposed in DETR [36]. First, we need to define the bounding box loss $\mathcal{L}_{box}$. It includes a scale-invariant generalized IoU loss [37] $\mathcal{L}_{iou}$ and a scale-dependent l_1 loss. Loss l_1 has different scales for large and small bounding boxes. To balance the loss of bounding boxes with various size, the bounding box loss is defined as $\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{iou}\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{l1}||b_i - \hat{b}_{\sigma(i)}||_1$. b_i indicates the ground-truth bounding box, and $\hat{b}_{\sigma(i)}$ is the predicted bounding box with index $\sigma(i)$ of N elements permutation S_N between candidate and ground-truth boxes.

The procedure of matching loss consists of two steps.

Finding Matching

The optimal matching result is defined as $\hat{\sigma} = \arg\min_{\sigma \in S_N} \sum_i^N \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$. The optimal assignment can be obtained by Hungarian algorithm as claimed in the original paper[36].

In addition to box loss $\mathcal{L}_{box}$, the model needs to consider the class prediction during matching procedure. The probability of target class c_i is defined as $\hat{p}_{\sigma(i)}(c_i)$. The matching cost is conveyed as:

$$\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) = -\mathbf{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}).$$

Noting that, $\mathcal{L}_{match}$ is only used for finding the optimal assignment. There is no log for the class probability term, it is claimed to have better balance with bounding box loss $\mathcal{L}_{box}$.

Loss Computation

The matching loss to update model parameter is computed based on the optimal bounding box assignment $\hat{\sigma}$ from previous step. It is defined in DETR as :

$$\mathcal{L}_{\text{Hungarian}}(y_i, \hat{y}_{\hat{\sigma}(i)}) = -\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}).$$

The overall loss is summed over all boxes: $\sum_{i=1}^N \mathcal{L}_{\text{Hungarian}}$.

2.4 In-depth Study of Three Foundational Methods

In this project, we delve into three pioneering methods that serve as the cornerstone of our baseline models: BEVFusion[4], DeepInteraction[5], and the Cross Modal Transformer (CMT)[6].

BEVFusion [4] extracts LiDAR and camera features in parallel and uses a fusion module to perform data fusion after generating the LiDAR and camera BEV features. DeepInteraction [5] enhances data fusion through feature interaction within both the encoder and decoder modules. Cross Modal Transformer (CMT) [6] utilizes additional position encoding to implicitly align the LiDAR and camera features.

A comprehensive analysis of these methods is conducted to explore their conceptual foundations and technical architectures.

2.4.1 BEVFusion

BEVFuion[4] performs data fusion by separately extracting features from camera and LiDAR data, utilizing an additional fusion module to align them within a unified BEV space.

Architectural Components

For image feature extraction, BEVFusion employs a Dual-Swin-Tiny Transformer[38] combined with FPNC mentioned in section 2.3.4 as the image backbone. Meanwhile, the LiDAR backbone for extracting point cloud features consists of HardSimple VFE (Voxel Feature Encoding), a Sparse Encoder, SECOND (Spatio-TEmporal Deep Convolutional Networks), and SECONDFPN, enhancing the detection capabilities through depth and spatial resolution.

a. Image Backbone

The Swin Transformer[39], a variant of vision transformers, introduces a hierarchical architecture using shifted windows to compute representations. This approach limits self-attention computation to non-overlapping local windows while enabling cross-window connections. It constructs image representations starting from small-sized patches. As processing advances to deeper Transformer layers, neighboring patches are merged, thereby forming a hierarchical representation that enhances the model's ability to capture both fine-grained and broader spatial features.

The Dual-Swin-Tiny Transformer[38] represents an integration of the Composite Backbone Network (CBNet) architecture with the Swin Transformer[39], aimed at constructing high-performance backbone networks for object detection without additional pre-training. CBNet operates by grouping multiple identical backbones in parallel, a strategy that enables the integration of both high-level and low-level features from these backbones. This design expands the receptive field that enhances the network's ability to efficiently perform object detection.

After the extraction of deep image features via the image backbone, the view projection module, as proposed in LSS[40], is employed to transform the 2D features into a 3D space. This transformation of a 2D feature point into a 3D voxel is accomplished by predicting the depth of image features and utilizing the camera's extrinsic parameters. Then a BEV Encoder Module used spatial to channel operation by stacking the dimensions of channel and height is applied to further encode the voxel feature into the BEV space feature.

b. LiDAR Backbone

The LiDAR data undergoes an initial transformation from point clouds to voxels using a hard voxelization process, specifically through the HardSimple Voxel Feature Encoding layer, as detailed in section 2.3.2. The voxels are then input into a Sparse Encoder, which extracts a sparse 3D feature map and delineates the spatial relationships within the data. The output from the Sparse Encoder is then processsed by the SECOND network[29], designed to extract the high-level spatial features via sparse convolution as mentioned in section 2.3.5. Finally, these features are routed to the SECONDFPN. SECONDFPN functions analogously to the traditional FPN but is optimized for handling 3D feature maps, to integrate information across different scales. The transformation from 3D voxel features into BEV features is done by reducing the height dimension.

c. Architecture

After extracting features from image and LiDAR separately into image and points BEV Feature, BEVFusion applies a dynamic fusion module utilizing channel attention module to select important fused features to concatenate to form the fused BEV feature. The fused feature is then fed into the detection head for decoding and making predictions on the bounding box and class of the objects.

The fusion module is defined as

$$\begin{aligned} \mathbf{F}_{fused} &= f_{adaptive}(f_{static}([\mathbf{F}_{Camera}, \mathbf{F}_{LiDAR}])) \\ f_{adaptive}(\mathbf{F}) &= \sigma(\mathbf{W} f_{avg}(\mathbf{F})) * \mathbf{F} \end{aligned} \tag{2.1}$$

The BEV features extracted from both camera and LiDAR sensors, denoted as $\mathbf{F}_{Camera}$ and $\mathbf{F}_{LiDAR}$ are initially concatenated along the channel dimension. Subsequently, a 3x3 convolution layer is employed to perform static channel and spatial fusion to reduce the channel dimension of the concatenated feature to to match the camera's channel dimension. the adaptive function involves multiplying the outcome of the static fusion with the result of a sigmoid function. This sigmoid function is applied to the product of a linear transformation matrix $\mathbf{W}$ and the global average pooling operation on the concatenated feature.

The overall architecture of BEVFusion is illustrated in Figure 2.4.1.

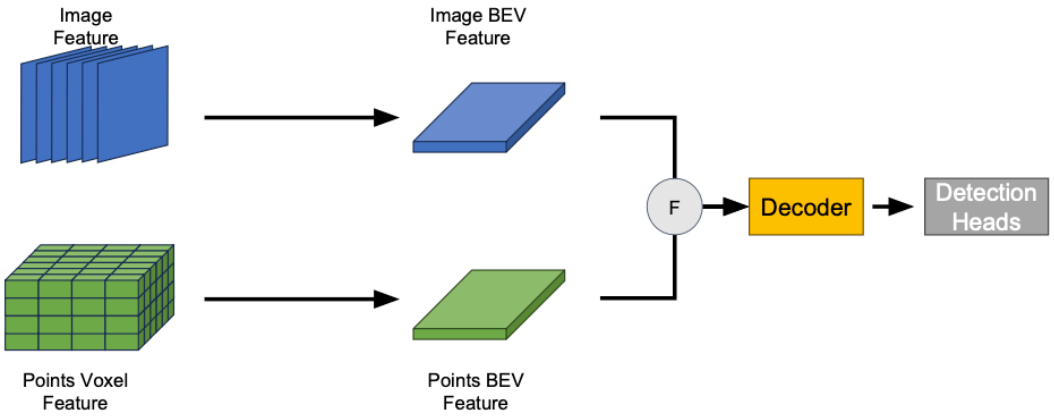

Figure 2.1: Architecture of BEVFusion, adpated from the original paper[4].

d. Detection Head

BEVFusion utilizes three popular detection heads in different categories, from anchor-based[10] to anchor-free-based[11] and transform-based[12].

On NuScenes data, it employed the transform-based detection head inherited from Transfusion[12].The object queries containing instance information are decoded into boxes and class labels with prediction on the center position, bounding box height, size, yaw angle and the velocity, and a per-class probability for the semantic classes.

Matching Loss

a. Matching Loss

BEVFusion[4] follows the design of matching loss used in Transfusion[12], it is defined as a weighted sum of classification, regression and IoU cost. Focal loss[41] is applied as the classification loss. The regression loss and IoU cost which are the same as the l_1 loss and IoU loss described in the section 2.3.8.

$$\begin{aligned} C_{match} &= \lambda_{cls}\mathcal{L}_{cls}(p, \hat{p}_{\sigma(i)}) + \lambda_{L1}\mathcal{L}_{reg}(b, \hat{b}_{\sigma(i)}) + \lambda_{iou}\mathcal{L}_{iou}(b, \hat{b}_{\sigma(i)}) \\ &= \lambda_{cls}(-\alpha_t(1-\hat{p}_{\sigma(i)})^{\gamma} log(\hat{p}_{\sigma(i)})) + \lambda_{l1}||b_i - \hat{b}_{\sigma(i)}||_1 + \lambda_{iou}\mathcal{L}_{iou}(b, \hat{b}_{\sigma(i)}) \end{aligned} \tag{2.2}$$

b. Matching cost for assigning boxes

For the matching cost used for assigning boxes, it follows the same design as DETR[36], using bipartite matching between the predictions and ground truth objects through the Hungarian algorithm.

$$\mathcal{L}_{\text{Hungarian}}(y_i, \hat{y}_{\hat{\sigma}(i)}) = -\log\hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}}\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)}) \tag{2.3}$$

Data Processing

BEVFusion performs no data augmentations on the original input but conduct BEV-space data augmentation after achieving the BEV features when comparing with state-of-art methods.

2.4.2 DeepInteraction

DeepInteraction[5] enhances data fusion through an attention-based feature interaction within its encoder module, while both point clouds and image representations are preserved in the decoder module. Within the decoder, multimodal predictive interaction layers are employed. These layers facilitate the utilization and mutual interaction between LiDAR points and camera representations, aiming to refine bounding box predictions.

Architectural Components

The feature extraction in the LiDAR branch of DeepInteraction is the same as BEV-Fusion. For the image branch, DeepInteraction employs ResNet-50[42] combined with FPN mentioned in section 2.3.4 as the image backbone. The ResNet-50 is initialized from Cascade Mask R-CNN[43] which is an instance segmentation model, it is first pretrained on COCO[44] and then nuImage[3].

a. Image Backbone

ResNet-50 [42] is a residual learning framework comprising 50 layers in total, divided into 5 stages. The main concepts employed in ResNet-50 are residual representations and residual networks using shortcut connections.

The residual representations utilized in the framework encode features via residual vectors with respect to a dictionary. By encoding residual vectors instead of original vectors, it becomes more effective for vector quantization. Shortcut connections are a method to add additional connections from the network input to the output, addressing the problem of vanishing gradients in the training process.

Stage 0 takes the input image with dimensions (3, 224, 224), applies a convolutional layer with a size of 7x7, followed by batch normalization, ReLU activation, and a max-pooling layer to reduce the dimension of the input to (64, 56, 56).

Two types of bottleneck architectures are used repeatedly in stages 1 to 4 as different methods to perform shortcut connections. The first type employs projection shortcuts to increase dimensions while keeping the other shortcuts as identity functions, ensuring that the input and output have the same channels. This can be viewed as the convolutional block. The second type consists of a stack of 3 layers, including 1x1, 3x3, and 1x1 convolutions, along with parameter-free identity shortcuts to deepen the network. This can be viewed as the identity block.

Stage 1 includes one convolutional block and two identity blocks, outputting features with dimensions of (256, 56, 56). Stage 2 comprises one convolutional block and three identity blocks, yielding features with dimensions of (512, 28, 28). Stage 3 employs one convolutional block and 5 identity blocks to produce features with dimensions

of (1025, 14, 14). Stage 4 utilizes one convolutional block and two identity blocks, resulting in features with dimensions of (2048, 7, 7).

b. LiDAR Backbone

The processing of LiDAR data in DeepInteraction follows the same pipeline as BEV-Fusion described in the section 2.4.1.

c. Architecture

DeepInteraction consists of two main components: one encoder, which performs multi-modal representational interaction, and one decoder, which performs multi-modal predictive interaction. Both the encoder and decoder employ a transformer-like structure. The encoder takes the extracted features of LiDAR and images from the backbones and outputs two refined representations. The decoder is used to enhance the 3D object detection of one modality conditioned on the other modality, enabling it to solve the set prediction problem for bounding boxes and object categories in 3D objects.

Encoder

The encoder comprises two types of encoding layers: multi-modal representational interaction (MMRI) layers and intra-modal representational learning (IML) layers.

The multi-modal representational interaction (MMRI) layer operates in two steps to exchange neighboring context in a bilateral cross-modal manner between the image perspective representation h_c and the LiDAR BEV representation h_p. First, this layer establishes pixel-to-pixel(s) correspondence between the two representations using depth completion and projection for the image to the LiDAR BEV mapping frame $M_{c \to p}$. Second, it employs projection from 3D LiDAR points to image coordinates based on camera intrinsics and extrinsics as the LiDAR BEV to image coordinate frame $M_{p \to c}$. Attention-based feature interaction is then applied to the image feature points with their cross-modality neighbors, and to the LiDAR BEV feature points with their cross-modality neighbors. The output representation can be expressed as $h_p^{c \to p}$ and $h_c^{p \to c}$ for the interacted LiDAR BEV and image perspective representation respectively.

The intra-modal representational learning (IML) layer applies local attention to the image feature points with their local k x k grid neighbors and to the LiDAR BEV feature points with their local k x k grid neighbors. The output representation can be expressed as $h_p^{p \to p}$ and $h_c^{c \to c}$ for the intra-modal LiDAR BEV and image perspective representation respectively.

The output of the multi-modal representational interaction (MMRI) layer and intra-modal representational learning (IML) layer, along with the original encoded feature representations of LiDAR and camera, are integrated using a feed-forward network and concatenation, as shown in 2.4. 'FFN' refers to the feed-forward network, and 'Concat' indicates element-wise concatenation.

$$
\begin{aligned}
h_p^{'} &= FFN(Concat(FFN(Concat(h_p^{p\to p}, h_p^{c\to p}), h_p)) \\
h_c^{'} &= FFN(Concat(FFN(Concat(h_c^{c\to c}, h_c^{p\to c}), h_c))
\end{aligned}
\tag{2.4}
$$

Decoder and Detection Head

The predictive interaction decoder utilizes a multi-modal predictive interaction layer (MMPI), which takes the object queries and bounding box predictions from the previous layer as inputs, enabling interaction between the intensified image representation h'_c and h'_p in the layer.

For MMPI on the image, interaction involves extracting N Region of Interest (RoI) features from the intensified image representation. This is achieved by projecting the 3D bounding boxes onto the image representation to obtain a 2D convex polygon and then determining the minimum axis-aligned circumscribed rectangle. A multi-modal predictive interaction operator maps the object queries into the parameters of a series of 1 x 1 convolutions, which are applied to the RoI features to update the object query.

For MMPI on LiDAR, the approach is similar, but the RoI features are obtained by projecting the 3D bounding boxes onto the LiDAR BEV representation and determining the minimum axis-aligned rectangle. The box size is enlarged by a factor of 2 to address the issue of tiny-scale objects in the BEV coordinate frame.

Queries are first initialized using heatmap, a LiDAR transformer is used to integrate positional embeddings of 3D point cloud data to involve spatial relationship between points in the point cloud. The decoder consists of multiple MMPI on image and MMPI on LiDAR blocks, with a feed-forward network appended to the output queries for each MMPI layer. This network is used to infer the locations, dimensions, orientations, and velocities of the 3D objects.

The detailed modules of the decoder are illustrated in Figure 2.4.2.

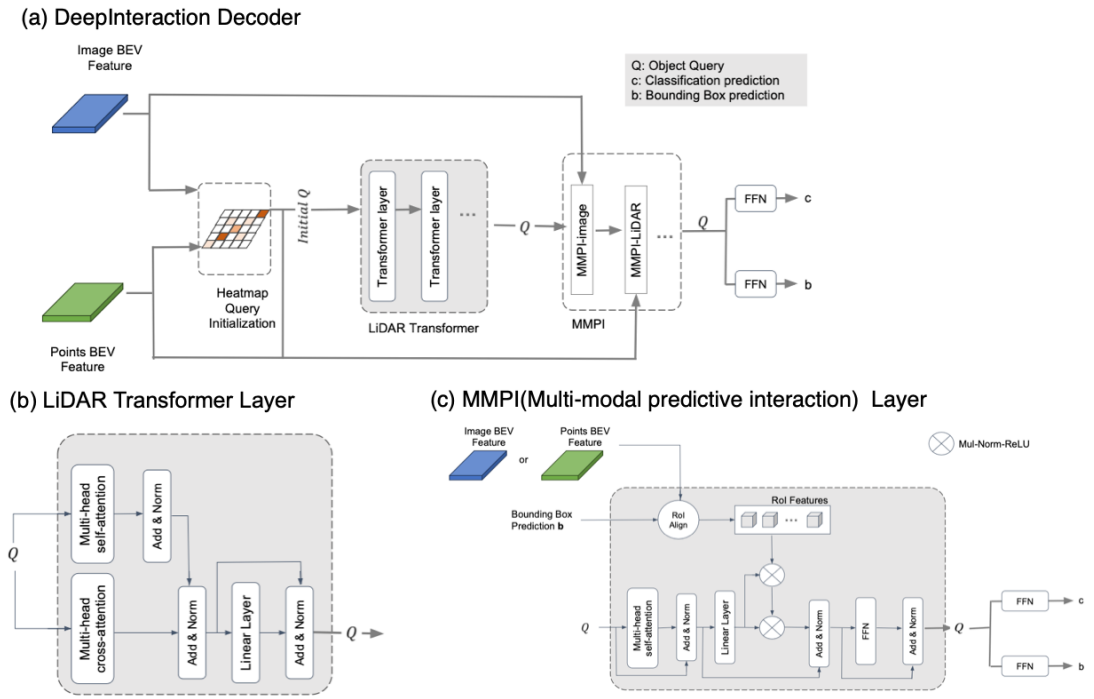

Figure 2.2: Modules of DeepInteraction Decoder, adapted from the original paper[5].

The overall architecture of DeepInteraction is illustrated in Figure 2.4.2.

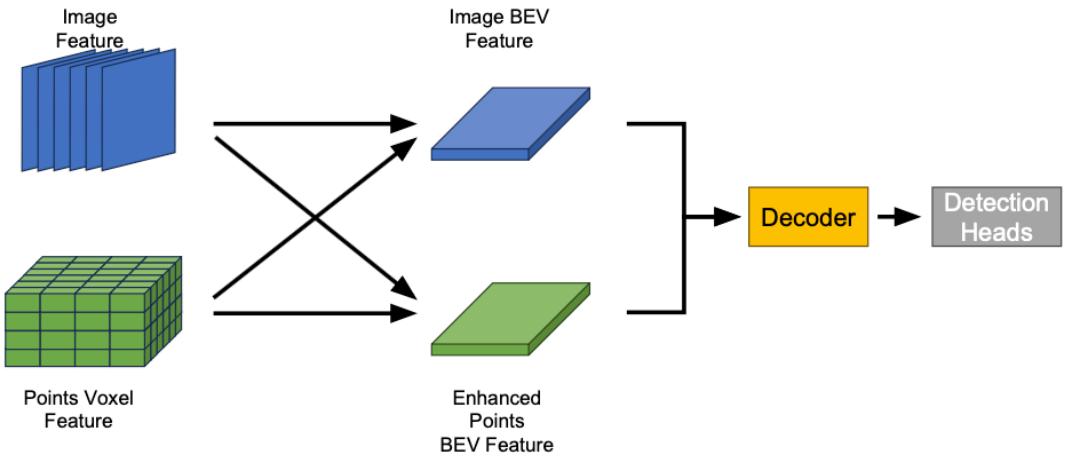

Figure 2.3: Simplified Architecture of DeepInteraction, adapted from the original paper[5].

Matching Loss

a. Matching Loss

DeepInteraction[5] follows the same design of matching loss used in Transfusion[12] as BEVFusion[4].

$$\begin{aligned} C_{match} &= \lambda_{cls}\mathcal{L}_{cls}(p, \hat{p}_{\sigma(i)}) + \lambda_{L1}\mathcal{L}_{reg}(b, \hat{b}_{\sigma(i)}) + \lambda_{iou}\mathcal{L}_{iou}(b, \hat{b}_{\sigma(i)}) \\ &= \lambda_{cls}(-\alpha_t(1-\hat{p}_{\sigma(i)})^{\gamma} log(\hat{p}_{\sigma(i)})) + \lambda_{l1}||b_i - \hat{b}_{\sigma(i)}||_1 + \lambda_{iou}\mathcal{L}_{iou}(b, \hat{b}_{\sigma(i)}) \end{aligned} \tag{2.5}$$

b. Matching cost for assigning boxes

DeepInteraction[5] uses the same matching cost for assigning boxes as DETR[36].

$$\mathcal{L}_{\text{Hungarian}}(y_i, \hat{y}_{\hat{\sigma}(i)}) = \sum_{i=1}^{N}[-\log\hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}}\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})].$$

Data Processing

DeepInteraction applies data augmentation, including random rotation, random scaling, random translation, and random horizontal flipping, to both LiDAR data and camera data. Additionally, a class-balanced resampling technique proposed in CBGS[45] is applied to balance the class distribution for nuScenes[3].

2.4.3 CMT

CMT[6] performs data fusion through the interaction between the object queries and multimodal features in the transformer decoder. It uses a coordinates encoding module (CEM) to add position encoding to image and LiDAR features and utilizes a position-guided query generator to incorporate the encoded position into the object queries.

Architectural Components

The feature extraction in the LiDAR branch of CMT is the same as BEVFusion. For the image branch, CMT employs VoVNet-V99-eSE[46] combined with CPFPN mentioned in section 2.3.4 as the image backbone.

a. Image Backbone

VoVNet-V99-eSE is a modification built upon VoVNet, incorporating an additional identity mapping and an Effective Squeeze-Excitation (eSE) module. The main concepts employed in this model are One-Shot Aggregation (OSA), Identity mapping, and eSE.

The OSA module consists of consecutive convolutional layers and aggregates the subsequent feature maps at once, as shown in Figure 2.4.3. This enables efficient capture of diverse receptive fields in terms of both accuracy and speed. VoVNet comprises multiple OSA modules in different stages.

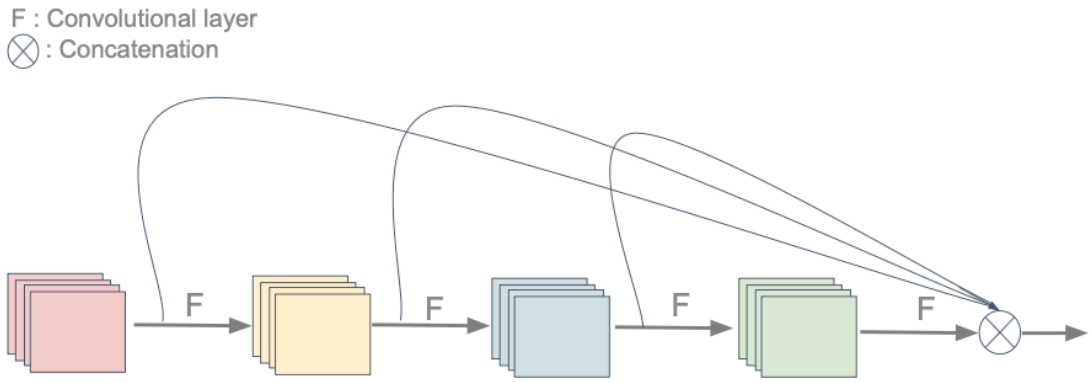

Figure 2.4: One-Short Aggregation module in VoVNet, adapted from the original paper[46].

Identity mapping is added to the OSA modules, creating a residual connection that connects the input path to the end of an OSA module. This allows the back-propagation of gradients through every OSA module in an end-to-end manner.

The eSE module is a representative channel attention method adopted in CNN architectures. It enhances the representation by explicitly modeling the inter-dependency between the channels of feature maps.

b. LiDAR Backbone

The processing of LiDAR data in CMT follows the same pipeline as BEVFusion described in the section 2.4.1.

c. Architecture

CMT utilizes a Coordinates Encoding Module to encode the 3D coordinates into the multi-modal tokens extracted from the image and LiDAR backbone. Subsequently, a position-guided Query Generator is applied to generate queries with position information. These queries interact with the multi-modal tokens in the transformer-based Decoder to predict the object class and 3D bounding boxes.

Coordinates Encoding Module

The Coordinates Encoding Module generates camera and BEV position encodings (PEs) and adds them to image tokens and point cloud tokens to encode the 3D position information. This allows the multi-modal tokens to align in 3D space implicitly.

The module can be defined by equation 2.6, where $P(u, v)$ is the 3D points set corresponding to the feature map $F(u, v)$ of each modality (the image feature for camera and the BEV feature for LiDAR), and (u, v) is the coordinate in the feature map. ψ represents a multi-layer perception (MLP) layer.

$$\Gamma(u, v) = \psi(P(u, v)) \tag{2.6}$$

(i) Coordinates Encoding for Image

The pixel of image feature F_{im} can be formulated as a series of points in the camera frustum coordinates as $\{p_k(u,v) = (u * d_k, v * d_k, d_k, 1)^T, k = 1, 2, ..., d\}$, where d is the number of points sampled along the depth axis.

The pixels of the image feature are first converted to the corresponding 3D points by calculation using transformation matrix $T^l_{c_i}$ from the i-th camera coordinate to the LiDAR coordinate and the intrinsic matrix $K_i \in 4 \times 4$ of the i-th camera, as shown in the equation 2.7.

$$p^{im}_k(u,v) = T^l_{c_i} K_i^{-1} p_k(u,v) \tag{2.7}$$

Then the position encoding for the image feature pixel (u, v) is formulated as

$$\Gamma_{im}(u,v) = \psi_{im}(\{p^{im}_k(u,v), k = 1, 2, ..., d\}) \tag{2.8}$$

(ii) Coordinates Encoding for Point Clouds

The point set P of BEV feature for LiDAR can be sampled along the Z-axis, and can be formulated as $p^{pc}_k(u,v) = (u * u_d, v * v_d, h_k, 1)$, where u_d and v_d is the size of each BEV feature grid and h_k is the height of k-th points with a default $h_0 = 0$. Only one point is along the height axis is sampled.

Then the position encoding for the BEV feature for LiDAR (u, v) is formulated as

$$\Gamma_{pc}(u,v) = \psi_{pc}(\{p^{pc}_k(u,v), k = 1, 2, ..., h\}) \tag{2.9}$$

Position-guided Query Generator

Queries are initialized with n anchor points sampled from a uniform distribution between $[0, 1]$. Linear transformation is applied to the anchor points to project them from 2D into 3D space.

The transformed 3D anchor points are then projected onto different modalities and encoded by the Coordinates Encoding Module. This encoding involves the summation of positional embeddings of anchor point sets projected onto both the BEV plane (A_{pc}) and the image plane (A_{im}), as illustrated in 2.10. Here, Γ_q represents the function of the position embedding. The resulting position embeddings are then added to the query content to generate the initial position-guided queries Q_0 as input to the decoder.

$$\Gamma_q = \psi_{pc}(A_{pc}) + \psi_{im}(A_{im}) \tag{2.10}$$

Transformer-based Decoder

CMT follows the design of the transformer decoder used in DETR[47]. Position-guided queries are employed for each decoder layer to interact with the multimodal tokens and update the representations. Additionally, two feed-forward networks (FFNs) are appended to the output queries for each decoder layer to infer the 3D bounding boxes and classes.

The overall architecture of CMT is illustrated in Figure 2.4.3.

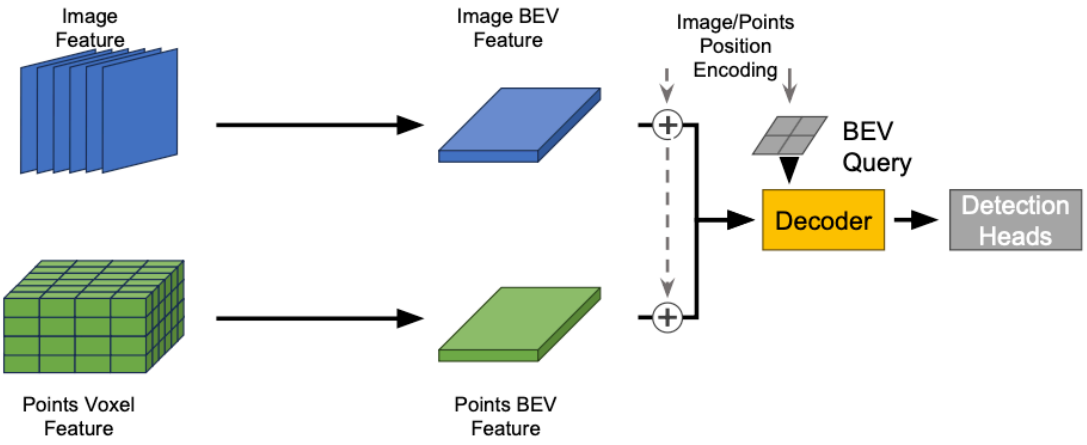

Figure 2.5: Simplified Architecture of CMT, adpated from the original paper[6].

Matching Loss

a. Matching Loss

CMT[6] follows the design of matching loss used in DETR[47] except the use of IoU cost, it is defined as a weighted sum of classification and regression cost. Focal loss[41] is applied as the classification loss. The regression loss is the same as the l_1 loss described in the section 2.3.8.

$$\begin{aligned} C_{match} &= \lambda_{cls}\mathcal{L}_{cls}(p, \hat{p}_{\sigma(i)}) + \lambda_{L1}\mathcal{L}_{reg}(b, \hat{b}_{\sigma(i)}) \\ &= \lambda_{cls}(-\alpha_t(1-\hat{p}_{\sigma(i)})^{\gamma} log(\hat{p}_{\sigma(i)})) + \lambda_{l1}||b_i - \hat{b}_{\sigma(i)}||_1 \end{aligned} \tag{2.11}$$

b. Matching cost for assigning boxes

The matching cost used for assigning boxes follows the same design as DETR[36], employing bipartite matching between the predictions and ground truth objects through the Hungarian algorithm. Unlike using the bounding box loss as a sum of regression and IoU cost, only regression loss is utilized.

$$\mathcal{L}_{\text{Hungarian}}(y_i, \hat{y}_{\hat{\sigma}(i)}) = \sum_{i=1}^{N}[-\log\hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}}\mathcal{L}_{L1}||b_i - \hat{b}_{\sigma(i)}||_1]$$

Data Processing

CMT utilizes the class-balanced resampling technique proposed in CBGS [45] during training for 20 epochs. It employs GT(ground truth) sample augmentation for the first 15 epochs and discontinues it for the remaining epochs.

3

Method

We elaborate the motivation and implementation of our method in this section. It mainly includes two modules, namely Global Position Encoding (GPE) and Information Enhanced Decoder (IED).

3.1 Global Position Encoding (GPE)

As we discuss in Section 2.4.3, CMT [6] proposed coordinate encoding for LiDAR and image data. However, we cannot merge it to our baseline models simply with similar encoder and decoder networks. Therefore, to explore the effect of coordinate encoding, we implement GPE modules and plug it into baseline models to have a fair performance comparison.

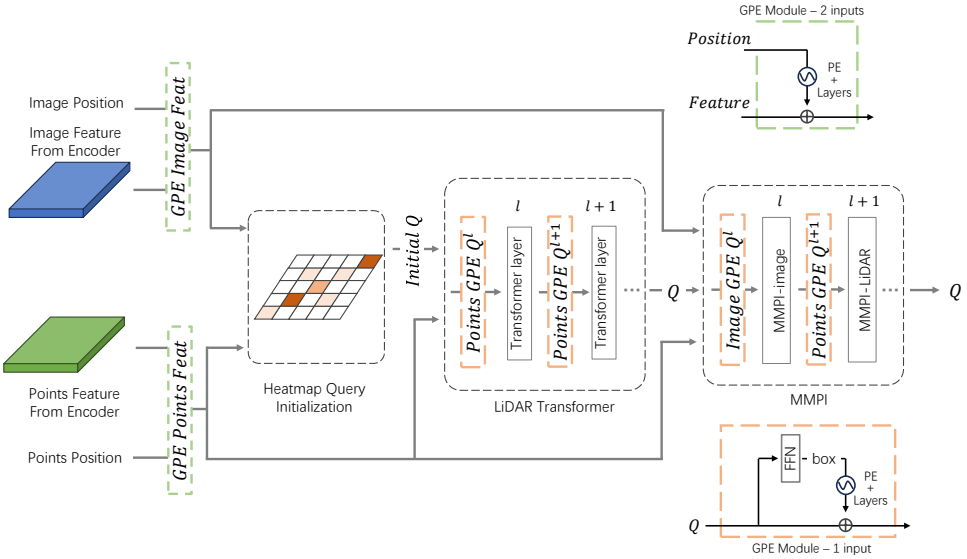

Figure 3.1: Decoder with Global Position Encoding. The base decoder structure is implemented in DeepInteraction [5]. We also introduce MMPI in Section 2.4.2. We merge the GPE modules (green and orange dash boxes) into baseline models with similar encoder and decoder networks.

3.1.1 BEV Location of LiDAR Feature

Since the LiDAR backbone network produces the feature of voxels without z-axis, it is natural to obtain the corresponding BEV position. Formally, V_i indicates the voxel with index i, its BEV position is conveyed as :

$$POS_{BEV}(V_i) = \text{Project}_{\text{LiDAR}\rightarrow\text{BEV}}(V_i)$$

3.1.2 BEV Location of Camera Feature

As described in DeepInteraction[5], the depth information of image pixels from points cloud is obtained by first transforming the coordinate, then projecting the pixels back to LiDAR coordinate. We note x_i, y_i, z_i as the points position in LiDAR coordinate and u_i, v_i, d_i are the position and depth of the corresponding pixel $Pixel_{V_i}$. We present the mapping process as follows:

$$\begin{aligned} u_i, v_i, d_i &= \text{Project}_{\text{LiDAR}\rightarrow\text{Camera}}(\text{x}_\text{i}, \text{y}_\text{i}, \text{z}_\text{i}) \\ POS_{BEV+z}(Pixel_{V_i}) &= \text{Project}_{\text{Camera}\rightarrow\text{BEV+z}}(u_i, v_i, d_i) \end{aligned}$$

3.1.3 Position Encoding Generation

First, we encode the obtained BEV positions of camera and LiDAR feature by sinusoidal position encoding as we introduced in Section 2.3.6. Then, several fully connected layers are followed to adapt to the dimensionality of camera and LiDAR features.

$$Feature_{PE} = \text{FC}(\text{PE}(POS_{BEV}))$$

It is worth noting that we use separate network to encode image and points position.

3.1.4 GPE Modules

We implement the GPE as illustrated in Figure 3.1.

Camera/LiDAR Feature GPE Module

To include position information for camera and LiDAR features, we first generate position encoding according to the process as above and add them to features element-wisely (green dash box in Figure 3.1).

Query Feature GPE Module

To align the positions across points and images, we also let the query features incorporate with position information (orange dash box in Figure 3.1). We obtain the center position of proposal bounding boxes via prediction heads firstly, then using a similar network to generate position encoding and merge with the query features element-wisely. It is noting that, since the query features are refined gradually through the whole model, we also need to update the PE in each step.

The parameters are shared for the GPE module that processes image-related features and the same for the module that deals with LiDAR-related features.

By modeling the BEV position as conveyed, we let the position information be consistent and fully interact for different modalities in the whole network.

3.2 Information Enhanced Decoder (IED)

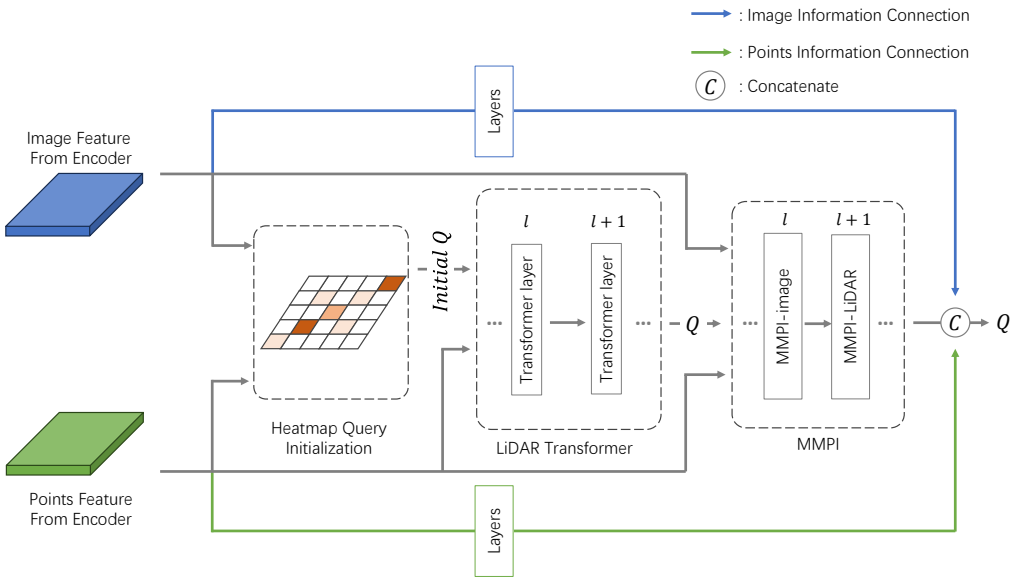

Figure 3.2: Information Enhanced Decoder Module. The base decoder structure is implemented in DeepInteraction [5].The IED module connects the output of encoder to the the output of original decoder directly.

The motivation of IED module (Figure 3.2) is to avoid information loss during the process of transforming the features in to BEV plane. IED enhances decoder by connecting the output from encoder to the last step of decoder directly. To have the same shape as query feature, we use several layers to adjust the dimensionality. We will provide the specific layer structure in experimental setting section.

3.2.1 RoI Feature Extraction

To align the output of encoder to the query feature, firstly we need to extract the feature at corresponding RoI positions.

$$RoIFeat_i^{Encoder} = \text{Pooler}(Feat^{Encoder}, Box_i)$$

Box_i is bounding box i predicted by RoI detection head with query feature in the previous step, $Feat^{Encoder}$ indicates the features from encoder and Pooler is the RoI Pooler [48] to extract features of Box_i.

3.2.2 Feature Concatenation

We indicate the query feature of the corresponding Box_i as $RoIFeat_i^Q$.

$$RoIFeat_i^{Q'} = \text{Concat}(RoIFeat_i^Q, RoIFeat_i^{Encoder})$$

We concatenate $RoIFeat_i^Q$ and $RoIFeat_i^{Encoder}$ together as the information enhanced feature $RoIFeat_i^{Q'}$. It will be used as input to the detection heads.

4

Experiments

In this section, we present the details of experiments, including dataset construction, baseline models and experimental settings. The experimental results and corresponding analysis are presented in the next chapter.

4.1 Data Sets

We utilize the nuScenes[3] public dataset to conducts experiments. It provides a rich surround view by means of comprehensive sensor data from 1 spinning LiDAR, 6 RGB cameras and five radars. We only use the data of LiDAR and cameras.

4.1.1 Sub-dataset of nuScenes split

To reduce the computation time for experiments, we build a sub-dataset instead of the full original dataset. Firstly, we keep the official splitting strategy named as "mini" dataset, which includes 8 and 2 scenes for training and validation sets respectively. Then we randomly select extra 32 training scenes for the training set.

For the validation set, we remove the overlapping scenes from all 150 validation ones. To summarise, we train all models on a set with 40 scenes, and validate their performance on validation set with other 148 scenes.

Comparing with the full training data of nuScenes, which consists of 850 scenes, the sub-dataset comprises approximately 5% of the full training data. To verify the reliability of using 5% of the training data for comparing different data fusion methods, we first employ a 10-scene validation set to compare the results of various experiments. These results are analyzed in section 5.2. Furthermore, to delve deeper into this comparison, we validate the models trained by the sub-dataset on the full validation set, which consists of 148 scenes as described above. The detailed scenes of the sub-dataset are listed in Appendix A.

4.1.2 Sub-dataset of geographical split

In addition to the original split of the nuScenes data, Adam et al. [49] proposed a geographically split method to divide samples according to geographical positions for unbiased evaluation. This split aims to address the potential data leakage issue in nuScenes that the same geographical position is re-visited multiple times across

the different sets of training, validation and test. With the proposed split method, the validation and test set have no overlapping geographical position as the training set.

We constructed a training set consisting of 40 scenes and a validation set of 210 scenes based on the geographical split. For the training set, 40 scenes were randomly selected. In forming the validation set, we combine the validation and test set of the geographical split, ensuring exclusion of scenes present in the previously created training set in Section 4.1.1 to ensure fair comparison. Additionally, scenes designated as the test set in the original nuScenes dataset were removed due to a lack of annotations. The detailed scenes of the geographical split for the sub-dataset are listed in Appendix A.

In this report, we refer to the official split of training, validation, and test scenes in the nuScenes dataset as nuScenes-split. Additionally, we use the term geo-split to denote the geographical split proposed by Adam et al[49].

4.2 Baseline Models

To analyze the contribution of various data fusion methods, we employ 3 baseline models (i.e., BS1, BS2, and BS3 described in this section). We let all these baseline models and our model have the same point clouds features from a 3D backbone network and image features from a 2D backbone network. We follow the same settings as in DeepInteraction[5] to configure backbone networks, as we described in Section 2.4.2.

In addition, we also use a similar decoder network as in DeepInteraction for these 3 baselines, thus we can have a fair comparison.

BS1. Parallel BEV Feature (PBF)

This baseline model reads the camera and LiDAR feature independently, which is similar to the encoder module of BEVFusion [4].

Worth noting that, to corporate image data and ensure all model using the same features given by a 3D backbone network, we project the point clouds to images first, such that we can assign the value of depth to camera features. However, as we discussed in Section 2.4, the depth information in BEVFusion is predicted by a network taking image data as input.

BS2. Interaction Enhanced (ITE)

We set the DeepInteraction[5] model as the 2nd baseline model. In this baseline, the image feature and cloud points features are interacted in encoder module, as we conveyed in Section 2.4.

BS3. Global Position Encoding (GPE)

Compared to BS2, we add the Global Position Encoding module as we described in Section 3.1.

Ours. Information Enhanced Decoder(IED)

Additionally to BS3, our own model implements the Information Enhanced Decode module as we proposed in Section 3.2.

4.3 Evaluation Metrics

To evaluate models' performance, we plan to use the following evaluation metrics that have been used in previous works [4]–[6], [27].

Mean Average Precision (mAP) The matching is defined as the 2D center distance on the ground plane. The matching average precision is obtained by integrating recall-precision curve for the recall and precision > 0.1. To cover a wider range of matches and reflect differences among models, the final index is computed by averaging the results with the match thresholds of 0.5,1,2,and 4 meters.

NuScenes Detection Score (NDS) NDS is calculated according to

$$NDS = \frac{1}{10}[5\text{mAP} + \sum(1 - \min(1, \text{TP\_score}))],$$

where TP_score includes 5 additional indexes : Average Translation Error (ATE), Average Scale Error (ASE), Average Orientation Error (AOE), Average Velocity Error (AVE), and Average Attribute Error (AAE).

4.4 Experimental Setting

4.4.1 Data Processing

We conduct the experiments with MMDetection3D[1] package. We follow the official procedure to conduct the pre-processing.

In terms of data augmentation methods, we use the same pipeline as the setting in DeepInteraction[5].

4.4.2 GPE Module Network Structure

Table 4.1: Network Layers of Image GPE

Layer	Parameter Setting
Linear	Input:128*3, Output:128*4
ReLU	
Linear	Input:128*4, Output:128*2
ReLU	
Linear	Input:128*2, Output:128*1

[1] `https://mmdetection3d.readthedocs.io/en/latest/`

Table 4.2: Network Layers of LiDAR GPE

Layer	Parameter Setting
Linear	Input:128*2, Output:128*4
ReLU	
Linear	Input:128*4, Output:128*2
ReLU	
Linear	Input:128*2, Output:128*1

4.4.3 IED Module Network Structure

Table 4.3: IED Network Layers

Layer	Parameter Setting
Conv2d	Input channel:128, Output channel:128, Kernel:7*7
Flatten	
Linear	Input:128, Output:128*2
Layer Normalization	
Linear	Input:128*2, Output:128

Noting that, we use separate parameters for image and points features.

4.4.4 Main Hyper-parameters

Table 4.4: Main Hyper-parameters Settings.

Parameter	Value Setting
Image Scale	(800, 448)
Point Clouds Range	[-54.0, -54.0, -5.0, 54.0, 54.0, 3.0]
Size of Voxel	[0.075, 0.075, 0.2]
Epochs	6
Batch Size	1 Sample
Learning Rate	Choose a start from [10,5,2], anneal to 0.0001
Matching Cost Weight:IoU	0.25
Matching Cost Weight:L1	0.25
Matching Cost Weight:Class	0.15
Loss Weight: Class	0.25
Loss Weight: L1	0.25
Loss Weight: Heatmap	1.0
Number of Proposals	200
Hidden Channel	128

5

Results Analysis

In this chapter, we presents the results of our experiments and analyze them to answer the following questions:

- **Q1.** Does an encoder with interaction module contributes to better detection results?
- **Q2.** How does global position encoding (GPE) affect the performance?
- **Q3.** Does an information enhanced decoder (IED) leads to a better detection quality?
- **Q4.** Does the presence of the same geographical information in both training and validation sets affect the performance of object detection tasks?

5.1 Experimental Results

The models involved in our experiments are as follows:

- BS1: Parallel BEV Feature (PBF).
- BS2: BS1+Interaction Enhanced (ITE).
- Global Position Encoding (GPE) module.
- Ours: Information Enhanced Decode (IED) module.

In addition to these models, we also conduct several ablation experiments to study the contribution of each module.

Model Performance of 5% sub-training dataset on 10-scenes validation Set.

We started with the smaller validation set first with the experiments, we performed the experiments of BS1 and BS2 on the 10-scene validation set to verify the reliability of using 5% of the training data in this project.

The results are shown in the Table 5.1.

Table 5.1: Performance Comparison of BS1 and BS2 on 10 valid-scenes.

Model	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BS1.BPF	0.6513	0.6894	0.3354	0.3236	0.3310	0.2587	0.1134
BS2.ITE	0.6775	0.7026	0.3340	0.3175	0.3228	0..2644	0.1234

The results of using 5% training data on 10 validation scenes are close to the scores presented in the paper of DeepInteraction[5], which with an mAP of 70.78 and NDS of 73.43 on the nuScenes test set, and an mAP of 69.85 and NDS of 72.63 on the nuScenes validation set.

Based on this comparison, we consider it is reasonable to assume that using 5% of the training data can effectively infer the performance of different data fusion methods when applied to the entire training dataset.

Hence, we further evaluated and analysed the methods on the full 148 validation scenes to demonstrate the validity of using 5% data for experimentation.

Q1. Modality Interaction in encoder Improves Detection Performance effectively.

Table 5.2: Performance Comparison of BS1 and BS2 on 148 valid-scenes.

Model	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BS1.BPF	0.6325	0.6788	**0.2829**	**0.2804**	**0.3151**	0.2995	0.1965
BS2.ITE	**0.6505**	**0.6861**	0.2869	0.2856	0.3361	**0.2895**	**0.1940**

By applying across-modality feature interaction, we observe great performance improvement in baseline ITE (Table 5.2). Except for mAP, ITE also brings benefits to ATE, ASE and AOE.

Q2. By plugging GPE modules, we observe performance gain for indices related to the position of detection objects compared to baseline models.

Table 5.3: Performance Comparison of BS1 + GPE and BS3(BS2+GPE) on 148 valid-scenes.

Model	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BS1.BPF	0.6325	**0.6788**	0.2829	**0.2804**	**0.3151**	**0.2995**	**0.1965**
BS1+GPE	**0.6439**	0.6714	**0.2801**	0.3081	0.3226	0.3771	0.2177
BS2.ITE	0.6505	**0.6861**	0.2869	0.2856	**0.3361**	**0.2895**	**0.1940**
BS2+GPE	**0.6516**	0.6807	**0.2826**	**0.2774**	0.3368	0.3394	0.2149

As illustrated in Table 5.3, GPE improves the performance both in BS1.BPF and BS2.ITE. Specifically, we observe significant improvement on mAP for BS1.

GPE is beneficial for encoding image and LiDAR data positions and can be considered as an implicit way to align and interact features across modalities. Since mAP

and ATE are the most relevant metrics for candidate location, their improvement is consistent with the GPE motivation.

Q3.1. The IED module is effective to have better results for BS1 and BS2, the improvement of model BS2 + IED is considerable.

Table 5.4: Performance Comparison of BS1 + IED and BS2 + IED on 148 valid-scenes.

Model	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BS1.BPF	0.6325	**0.6788**	0.2829	**0.2804**	**0.3151**	**0.2995**	0.1965
BS1+IED	**0.6359**	0.6760	**0.2822**	0.2956	0.3420	0.3042	**0.1949**
BS2.ITE	0.6505	0.6861	**0.2869**	**0.2856**	0.3361	**0.2895**	**0.1940**
BS2+IED	**0.6593**	**0.6872**	0.2889	0.2940	**0.3300**	0.3128	0.1988

1.With IED module, BS1 gains better mAP. During the generation of BEV feature, there could be information loss. For example, points features' depth information and image features' texture information could be damaged when they are transformed into flat BEV features. Our IED provides direct connection between encoder and each head of decoder, which supplements information for specific prediction tasks.

2.For the experiment BS2+IED, we observed improvement both for mAP and NDS. We also notice that, the improvement of mAP is greater than when we plug IED to BS1. It shows that when we apply the IDE module, the interaction feature can bring more benefits.

Q3.2. The combination of GPE and IED provides no additional benefit.

Table 5.5: Performance of the Combination of GPE and IED on 148 valid-scenes.

Model	mAP↑	NDS↑	Model	mAP↑	NDS↑
BS1.BPF	0.6325	**0.6788**	BS2.ITE	0.6505	0.6861
BS1+GPE	**0.6439**	0.6714	BS2+GPE	0.6516	0.6807
BS1+IED	0.6359	0.6760	BS2+IED	**0.6593**	**0.6872**
BS1+GPE+IED	0.6395	0.6722	BS2+GPE+IED	0.6586	0.6833

1. For BS1, BS1 + GPE achieves the best mAP, while BS1 alone has the highest NDS. IED module and the combination GPE+IED dose not bring improvement. As we discussed in Section 5.3, GPE could be regarded as implicit alignment and interaction for multi-modality features. Without interaction in encoder, IED itself does not interact the multi-modality features. It could be the reason that IED module does not improve the model BS1 + GPE.

2. With interaction in encoder, BS2 + IED gains best result for both mAP and NDS. The model BS2 + GPE + IED does not have further improvement. Once the image and points features are interacted in the encoder, the benefit of GPE is weakened. That is also the reason why GPE is more useful for BS1 than BS2.

On the other hand, the interacted features bring additional useful information for prediction tasks. IED module helps to remain and transmit that information from encoder to prediction heads directly. Thus, the BS + IED module along has the best result.

Q4. Investigating on whether the presence of the same geographical information in both training and validation sets affect the performance of object detection tasks.

Firstly, it's essential to note that due to the utilization of the geographical split, the training and validation data differ from those used in previous experiments. Therefore, direct comparison of scores with results from prior experiments is not feasible. We conducted experiments with BS1 and BS2 trained on 5% of the training data from both the original split and the geographical split.

Using 5% of the training data from the nuScenes split allows us to assess the model's performance when the same geographical information appears in both the training and validation sets. This allows us to test the potential data leakage in the nuScenes dataset, as inspired by Adam et al. [49].

On the other hand, training with 5% of the training data from the geographical split ensures that the training and validation datasets are isolated both in terms of data and geographical information. This approach ensures that the validation set remains unseen to the model.

To facilitate comparison, we also used the trained weights of DeepInteraction[5] on the nuScenes full training set, which corresponds to our BS2 ITE, and validated it on the 210-scenes validation set of geographical split.

In Table 5.6, nuScenes-split is used to refer to the official split of training, validation, and test scenes in the nuScenes dataset and the term geo-split is used to denote the geographical split proposed by Adam et al. [49]. The selection of training scenes of the subset of these two splits is described in Section 4.1.

Table 5.6: Performance of BS1 BPF and BS2 ITE trained on 5% training set with nuScenes split and geographical split, BS2 ITE trained on full nuScenes training set and validated on 210-scenes validation set of geographical split.

Model	Training Set	mAP	NDS
BS1.BPF	nuScenes-split (40 scenes)	0.7649	0.7701
BS2.ITE	nuScenes-split (40 scenes)	**0.7739**	**0.7742**
BS1.BPF	geo-split (40 scenes)	0.7613	0.7591
BS2.ITE	geo-split (40 scenes)	0.7684	0.7611
BS2.ITE	nuScenes-split (full 850 scenes)	0.8382	0.8000

Based on the results, we can infer that the inclusion of the same geographical information in both the training and validation sets does have some impact on the model's performance, resulting in slightly higher values for mAP and NDS score. In the study by Adam et al.[49], they observed a significant drop in model performance

when employing a geographical split in the task of online mapping. However, in our project, where object detection serves as the downstream task, it appears that the data leakage issue caused by geographical information affects but not greatly on the model's performance. Additionally, with the new geographical split, BS2 ITE still exhibits better performance.

In addition, when comparing the results attained by BS2 ITE trained on 40 scenes with those achieved by BS2 ITE trained on the full 850 scenes of the nuScenes training set, the difference in mAP does not exceed 8%, and in NDS does not exceed 4%. This further underscores the potential of using a small amount of data, suggesting it may be sufficient to train a model with satisfactory performance for object detection tasks.

5.2 Case Analysis

In this section, we select several samples to illustrate differences in predictions of 3 models: BS1.BPF, BS2.ITE and BS2+IED.

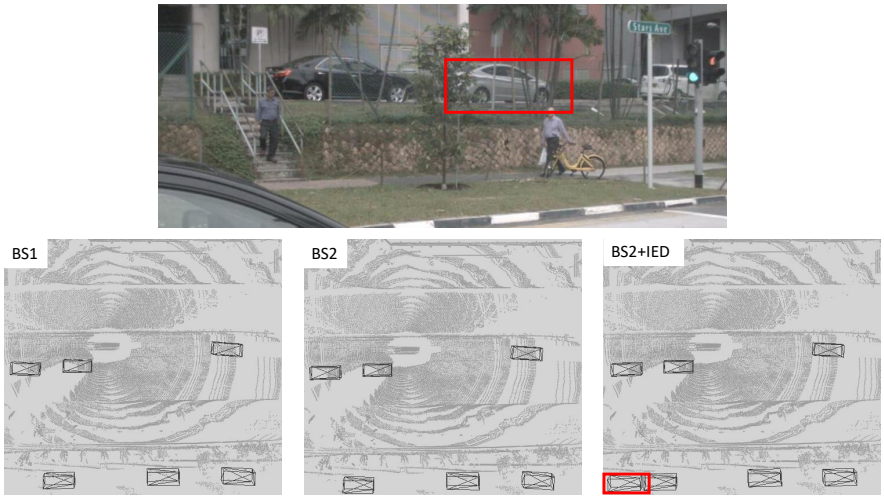

Figure 5.1: Detection Result Comparison Case 1

In Figure 5.1, BS2+IED detects the vehicle parking at the roadside in left-front, which is not recognised by BS1 and BS2.

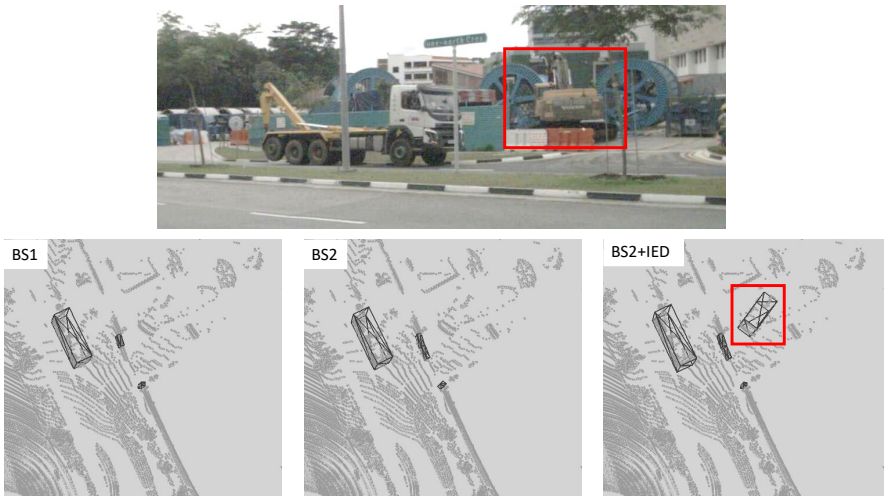

Figure 5.2: Detection Result Comparison Case 2

The machinery stopped behind the road barrier in Figure 5.2, which is not detected by BS1 and BS2.

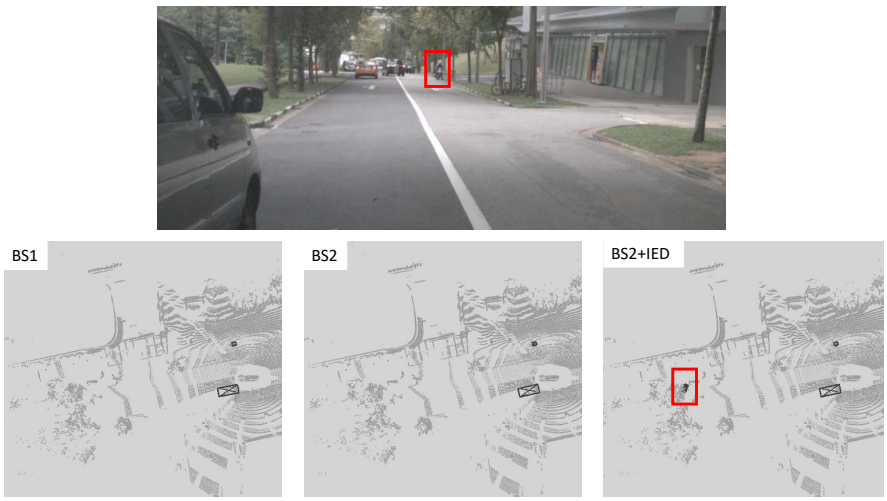

Figure 5.3: Detection Result Comparison Case 3

A motorcycle is located in front in Figure 5.3, which is detected by BS2+IED but not captured by other two models.

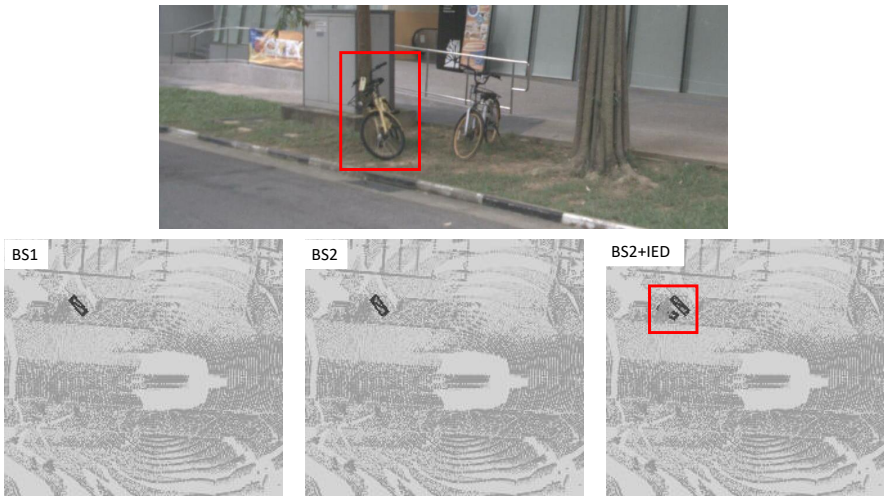

Figure 5.4: Detection Result Comparison Case 4

In Figure 5.4, two bicycles park in the front-right side. BS1 and BS2 only recognise one of them, while BS2+IED detects both.

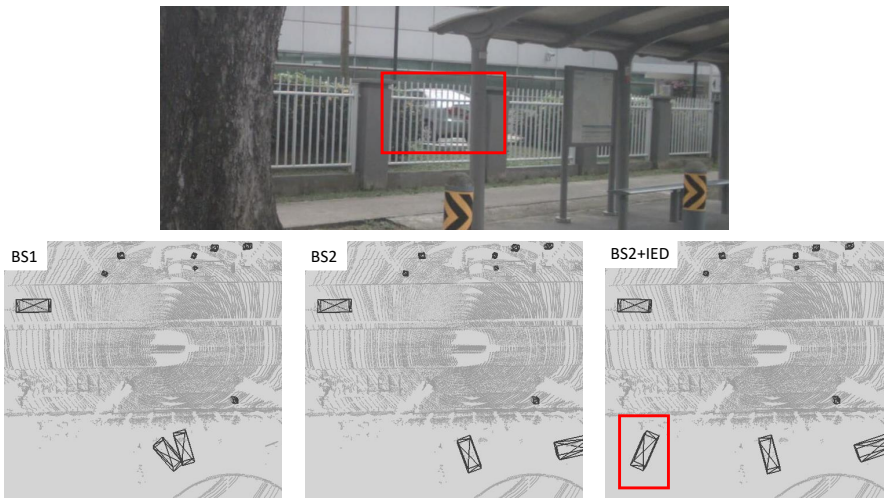

Figure 5.5: Detection Result Comparison Case 5

One vehicle parks behind wall fence in Figure 5.5. BS2+IED successfully detects the car, but neither BS1 nor BS2 identify it.

6

Discussion

This chapter discusses the limitations of this project, including risks, ethical considerations, and possibilities for future work.

6.1 Limitations

6.1.1 Influence of Sub-dataset

The complete nuScenes[3] public dataset comprises 850 scenes for training, 150 scenes for validation and testing respectively. Due to constraints in computational resources, we opted to create a smaller subset consisting of 40 scenes for training, as elaborated in Section 4.1. The specific scenes utilized in this subset are detailed in Appendix A.

Compared to the full 850 scenes, a training set of 40 scenes is a significantly smaller fraction. Consequently, there exists a possibility of inferior metric values when training models on this reduced dataset, as opposed to training with the entire dataset. To address this concern, we initially employed the smaller validation set to ascertain the reliability and representativeness of utilizing only 5% of the training data. Our findings suggest that this approach is reasonable and yields results indicative of the performance of various data fusion methods.

Subsequently, when evaluating the methods trained using the 5% dataset on the full set of 148 validation scenes, we observed a slight decrease in evaluation metrics. Nevertheless, these metrics remain comparable to those of the original DeepInteraction[5] model.

However, the utilization of this sub-dataset raises an important question: Is it sufficient to use a smaller dataset to train a model capable of handling 3D object detection in autonourmous vehicles? We will delve further into this possibility in Section 6.4, Future work.

6.1.2 Limited Fusion Paradigms

Our project focuses on exploring intermediate fusion, particularly focusing on feature-level data fusion paradigms. We take inspiration from three methods, BEVFusion[4], DeepInteraction[5], and the Cross Modal Transformer (CMT)[6] to design our experiments on the comparison of different feature-level data fusion approaches.

Throughout our project, we conducted four experiments to evaluate various feature-level data fusion methods. The first method, BS1 Parallel BEV Feature (PBF) extracts camera and LiDAR feature independently. BS2 Interaction Enhanced(ITE) is the same as DeepInteraction[5], the image and cloud points features are interacted in the encoder module. BS3 Global Position Encoding, incorporates additional position information into both the encoded and query features. The Information Enhanced Decoder leverages the information from encoded image and points cloud features to the decoder.

Our experiments primarily focused on examining how the interaction between features of different modalities, as well as the incorporation of extra position information and encoded features, can enhance the performance of data fusion methods for 3D object detection tasks. Due to time constraints, we did not delve deeper into fusion paradigms such as early-level (data-level) or late fusion (results-level) in this project. Analysis of these fusion paradigms is deferred to our future research.

6.1.3 Influence of Pretrained Backbone Networks

DeepInteraction [5] employes a pretrained image backbone network, ResNet-50 that is initialized from the instance segmentation model Cascade Mask R-CNN[43], which was pretrained on COCO[44] and nuImage[3], the related image datasets for autonomous driving provided by nuScenes. It is evident that the pretrained backbone impacts the model's performance for object detection. However, in this project, we maintain the same setting and do not conduct additional experiments to further investigate their influence.

6.1.4 Complex Architecture of Transformer

In this project, we employed several transformer-like architectures in our experiments, ranging from the encoder that utilizes self-attention and cross-attention mechanisms to interact with image and point cloud features, to the LiDAR transformer and the decoder layers. Transformers have demonstrated remarkable efficacy across various tasks in machine learning, however, they are much more complicated than simple neural networks. This complexity makes it challenging to precisely explain the impact of transformer architecture on the performance of data-fusion methods. Within the scope of this project, we were unable to conduct studies to examine the transformative power of transformers, we will explore potential ways of such investigations in Section 6.4, Future work.

6.2 Risk and Ethical Aspects

(1) Privacy Concerns

The data captured by sensors for autonomous driving may intrude upon individuals' privacy. According to nuScenes, they used the output of the object detectors to blur

faces and license plates in the images[1]. Hence the data is anonymized, adhering to privacy regulations.

(2) Bias in Data and Models

Adam et al. [49] suggested the potential issue of data leakage in nuScenes. NuScenes employs the same geographical positions across different sets of training, validation, and test data, potentially exposing the test set to data from the training and validation sets. This could introduce biases and result in partial model predictions. To mitigate this issue, we conducted a geographical split to assess the performance of the data fusion methods and determine whether biases in the data significantly impact the models. Interestingly, the results obtained from this geographical split were similar to those from the original split. This suggests that the nature of the object detection task may not be greatly influenced by geographical position, thereby minimizing the impact of potential biases.

6.3 Future Work

Given the experiments and results achieved in this project, more work can be done to further explore the data fusion methods.

First of all, in our experiments, we used a small amount of data for training, about 5% of the full nuScenes training set (40 out of 850 scenes). From the results we achieved using this 5% data on the full 148 validation set, for the Mean Average Precision (mAP), all of our experiments achieved a score over 0.63 and some of the experiments (BS2 ITE, BS2 + GPE, BS2 + IED) achieved a score over 0.65. For the NuScenes Detection Scores (NDS), all of our experiments achieved an NDS over 0.67, BS2 ITE and BS2 + GPE gains an NDS over 0.68.

Comparing these scores with the original DeepInteraction[5] trained on the full training set, which has an mAP of 69.85 and NDS of 72.63 on the nuScenes validation set, we can conclude that the difference in mAP is not larger than 0.05 (5%) and the difference in NDS is not larger than 0.03 (3%). However, these minor difference involves the use of additional 810 scenes of training data, which is a much larger data set. It seems that the use of huge amount of training data do not contribute to a significant improve on the performance, then it may possible that small amount of data is enough to train a model that work well for multimodal data fusion for the task of object detection in autonomous driving.

In this project, we only used nuScenes data, to verify the assumption of small amount of data, in future work, more autonomous driving dataset like KITTI[50] and Waymo[51] should be used and to experiment with smaller training subset.

Apart from further experiment with the amount of training data, we can explore more fusion paradigms like early-level(data-level) fusion and late fusion (results-level) to complement the comparison of more data fusion methods.

[1] `https://www.nuscenes.org/`

Besides, as we mentioned in the limitation, we employed transformer-like architectures in our experiments. Due to the complex nature of transformer, it's challenging to examine the impact of transformer architecture on the performance of data-fusion methods. In future experiment, we can replace the transformer component to simpler structure like deep neural network to examine how much transformer can affect the performance of the methods.

In this project, we adopted the same setting as DeepInteraction[5] for the pretrained backbone network to extract the image features. However, the impact of training the nuImage[3] data of the pretrained backbone on the model's performance remains unknown. In future research, we plan to explore this aspect by removing the nuImage part from the pretraining process and instead utilize the nuScenes data to train and update parameters of the image backbone. This investigation will allow us to assess the influence of the pretrained backbone on the model's performance.

7

Conclusion

In this project, we mainly explore camera and LiDAR data fusion methods for BEV perception. We begin our project by summarizing recent researches and analyzing their motivations. Then we look deeply into the techniques related to BEV perception tasks. To evaluate and analyze the performance of various models quantitatively, we also choose three foundational methods as baselines: BEVFusion[52], DeepInteraction[5] and CMT[6].

Moreover, to have fair comparison we propose our own method of Global Position Encoding (GPE) module and novel Information Enhanced Decoder (IED) module in Method Section. We conduct numerous experiments to evaluate the baselines and our methods. The Result Analysis Section illustrates their metric results and specific examples. We also state our interpretation of the effect of different modules. The experimental results show that the combination of DeepInteraction[5] and our GPE module further improves the performance on our dataset.

Since we build our own dataset, which is a subset of full nuSences dataset, we also investigate the effect of geographical property on detection tasks.

In Discussion Section, we summarize our considerations on the limitations of this project and potential further research directions.

Bibliography

[1] Y. Ma, T. Wang, X. Bai, *et al.*, "Vision-centric BEV perception: A survey," *CoRR*, vol. abs/2208.02797, 2022. DOI: 10.48550/ARXIV.2208.02797. arXiv: 2208.02797. [Online]. Available: https://doi.org/10.48550/arXiv.2208.02797.

[2] H. Li, C. Sima, J. Dai, *et al.*, "Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[3] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, *Nuscenes: A multimodal dataset for autonomous driving*, 2020. arXiv: 1903.11027 [cs.LG].

[4] T. Liang, H. Xie, K. Yu, *et al.*, *Bevfusion: A simple and robust lidar-camera fusion framework*, 2022. arXiv: 2205.13790 [cs.CV].

[5] Z. Yang, J. Chen, Z. Miao, W. Li, X. Zhu, and L. Zhang, "Deepinteraction: 3d object detection via modality interaction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1992–2005, 2022.

[6] J. Yan, Y. Liu, J. Sun, *et al.*, "Cross modal transformer: Towards fast and robust 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 268–18 278.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] T. Liang, X. Chu, Y. Liu, *et al.*, "Cbnet: A composite backbone network architecture for object detection," *IEEE Transactions on Image Processing*, vol. 31, pp. 6893–6906, 2022.

[9] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2403–2412.

[10] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[11] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[12] X. Bai, Z. Hu, X. Zhu, *et al.*, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1090–1099.

[13] Y. Cui, R. Chen, W. Chu, *et al.*, "Deep learning for image and point cloud fusion in autonomous driving: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 722–739, 2022. DOI: 10.1109/TITS.2020.3023541.

[14] D. Feng, C. Haase-Schütz, L. Rosenbaum, *et al.*, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2021. DOI: 10.1109/TITS.2020.2972974.

[15] Y. Wang, Q. Mao, H. Zhu, *et al.*, "Multi-modal 3d object detection in autonomous driving: A survey," *International Journal of Computer Vision*, pp. 1–31, 2023.

[16] D. Katare, D. S. Noguero, S. Park, N. Kourtellis, M. Janssen, and A. Y. Ding, *Analyzing and mitigating bias for vulnerable classes: Towards balanced representation in dataset*, 2024. arXiv: 2401.10397 [cs.CV].

[17] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," *IEEE/CVF conference on computer vision and pattern recognition*, pp. 4604–4612, 2020.

[18] Z. Chen, Z. Li, S. Zhang, L. Fang, Q. Jiang, and F. Zhao, "Autoalignv2: Deformable feature aggregation for dynamic multi-modal 3d object detection," *CoRR*, vol. abs/2207.10316, 2022. DOI: 10.48550/ARXIV.2207.10316. arXiv: 2207.10316. [Online]. Available: https://doi.org/10.48550/arXiv.2207.10316.

[19] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "Futr3d: A unified sensor fusion framework for 3d detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 172–181.

[20] Y. Li, L. Fan, Y. Liu, *et al.*, *Fully sparse fusion for 3d object detection*, 2023. arXiv: 2304.12310 [cs.CV].

[21] H. Cai, Z. Zhang, Z. Zhou, Z. Li, W. Ding, and J. Zhao, "Bevfusion4d: Learning lidar-camera fusion under bird's-eye-view via cross-modality guidance and temporal aggregation," *CoRR*, vol. abs/2303.17099, 2023. DOI: 10.48550/ARXIV.2303.17099. arXiv: 2303.17099. [Online]. Available: https://doi.org/10.48550/arXiv.2303.17099.

[22] H. Zhao, Q. Zhang, S. Zhao, Z. Chen, J. Zhang, and D. Tao, *Simdistill: Simulated multi-modal distillation for bev 3d object detection*, 2024. arXiv: 2303.16818 [cs.CV].

[23] S. Wang, H. Caesar, L. Nan, and J. F. P. Kooij, "Unibev: Multi-modal 3d object detection with uniform BEV encoders for robustness against missing sensor modalities," *CoRR*, vol. abs/2309.14516, 2023. DOI: 10.48550/ARXIV.2309.14516. arXiv: 2309.14516. [Online]. Available: https://doi.org/10.48550/arXiv.2309.14516.

[24] H. Wang, H. Tang, S. Shi, *et al.*, "Unitr: A unified and efficient multi-modal transformer for bird's-eye-view representation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6792–6802.

[25] Z. Wu, G. Chen, Y. Gan, L. Wang, and J. Pu, *Mvfusion: Multi-view 3d object detection with semantic-aligned radar and camera fusion*, 2023. arXiv: 2302.10511 [cs.CV].

[26] Y. Gao, C. Sima, S. Shi, S. Di, S. Liu, and H. Li, *Sparse dense fusion for 3d object detection*, 2023. arXiv: 2304.04179 [cs.CV].

[27] C. Hu, H. Zheng, K. Li, *et al.*, "Fusionformer: A multi-sensory fusion in bird's-eye-view and temporal consistent transformer for 3d object detection," *CoRR*, vol. abs/2309.05257, 2023. DOI: 10.48550/ARXIV.2309.05257. arXiv: 2309.05257. [Online]. Available: https://doi.org/10.48550/arXiv.2309.05257.

[28] Y. Zhou and O. 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*, 2018, pp. 4490–4499.

[29] Y. Zhou, P. Sun, Y. Zhang, *et al.*, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," in *Conference on Robot Learning*, PMLR, 2020, pp. 923–932.

[30] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: 1511.08458 [cs.NE].

[31] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[32] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[33] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[35] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[36] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, Springer, 2020, pp. 213–229.

[37] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 658–666.

[38] T. Liang, X. Chu, Y. Liu, *et al.*, "Cbnet: A composite backbone network architecture for object detection," *IEEE Transactions on Image Processing*, vol. 31, pp. 6893–6906, 2022, ISSN: 1941-0042. DOI: 10.1109/tip.2022.3216771. [Online]. Available: http://dx.doi.org/10.1109/TIP.2022.3216771.

[39] Z. Liu, Y. Lin, Y. Cao, *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. arXiv: 2103.14030 [cs.CV].

[40] J. Philion and S. Fidler, *Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d*, 2020. arXiv: 2008.05711 [cs.CV].

[41] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, *Focal loss for dense object detection*, 2018. arXiv: 1708.02002 [cs.CV].

[42] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[43] Z. Cai and N. Vasconcelos, *Cascade r-cnn: High quality object detection and instance segmentation*, 2019. arXiv: 1906.09756 [cs.CV].

[44] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV].

[45] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, *Class-balanced grouping and sampling for point cloud 3d object detection*, 2019. arXiv: 1908.09492 [cs.CV].

[46] Y. Lee and J. Park, *Centermask : Real-time anchor-free instance segmentation*, 2020. arXiv: 1911.06667 [cs.CV].

[47] Y. Wang, X. Zhang, T. Yang, and J. Sun, *Anchor detr: Query design for transformer-based object detection*, 2022. arXiv: 2109.07107 [cs.CV].

[48] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[49] A. Lilja, J. Fu, E. Stenborg, and L. Hammarstrand, *Localization is all you evaluate: Data leakage in online mapping datasets and how to fix it*, 2024. arXiv: 2312.06420 [cs.CV].

[50] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[51] P. Sun, H. Kretzschmar, X. Dotiwalla, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[52] Z. Liu, H. Tang, A. Amini, *et al.*, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 2774–2781.

A

Detailed Split of Data set

This chapter lists the detailed scenes of nuScenes data used in this project.

A.1 Split based on the original NuScenes split

A.1.1 Training Set(40 scenes)

The scenes used for the training set are:

'scene-0023', 'scene-0393', 'scene-0578', 'scene-0710', 'scene-0893', 'scene-0061', 'scene-0398', 'scene-0642', 'scene-0713', 'scene-0978', 'scene-0066', 'scene-0471', 'scene-0646', 'scene-0746', 'scene-1049', 'scene-0127', 'scene-0472', 'scene-0655', 'scene-0757', 'scene-1077', 'scene-0134', 'scene-0504', 'scene-0665', 'scene-0796', 'scene-1093', 'scene-0149', 'scene-0512', 'scene-0683', 'scene-0819', 'scene-1094', 'scene-0377', 'scene-0553', 'scene-0701', 'scene-0853', 'scene-1100', 'scene-0385', 'scene-0574', 'scene-0704', 'scene-0870', 'scene-1107'.

A.1.2 Small Validation Set(10 scenes)

The scenes used for the small validation set are:

'scene-0103', 'scene-0344', 'scene-0523', 'scene-0552', 'scene-0775', 'scene-0795', 'scene-0916', 'scene-0922', 'scene-0966', 'scene-1072'.

A.1.3 Validation Set(148 scenes)

The scenes used for the validation set are:

'scene-0003','scene-0012','scene-0013','scene-0014','scene-0015','scene-0016','scene-0017', 'scene-0018','scene-0035','scene-0036','scene-0038','scene-0039','scene-0092','scene-0093', 'scene-0094','scene-0095','scene-0096','scene-0097','scene-0098','scene-0099','scene-0100', 'scene-0101','scene-0102','scene-0103','scene-0104','scene-0105','scene-0106','scene-0107', 'scene-0108','scene-0109','scene-0110','scene-0221','scene-0268','scene-0269','scene-0270', 'scene-0271','scene-0272','scene-0273','scene-0274','scene-0275','scene-0276','scene-0277', 'scene-0278','scene-0329','scene-0330','scene-0331','scene-0332','scene-0344','scene-0345', 'scene-0346','scene-0519','scene-0520','scene-0521','scene-0522','scene-0523','scene-0524', 'scene-0552','scene-0554','scene-0555','scene-0556','scene-0557','scene-0558','scene-0559',

'scene-0560','scene-0561','scene-0562','scene-0563','scene-0564','scene-0565','scene-0625', 'scene-0626','scene-0627','scene-0629','scene-0630','scene-0632','scene-0633','scene-0634', 'scene-0635','scene-0636','scene-0637','scene-0638','scene-0770','scene-0771','scene-0775', 'scene-0777','scene-0778','scene-0780','scene-0781','scene-0782','scene-0783','scene-0784', 'scene-0794','scene-0795','scene-0797','scene-0798','scene-0799','scene-0800','scene-0802', 'scene-0904','scene-0905','scene-0906','scene-0907','scene-0908','scene-0909','scene-0910', 'scene-0911','scene-0912','scene-0913','scene-0914','scene-0915','scene-0916','scene-0917', 'scene-0919','scene-0920','scene-0921','scene-0922','scene-0923','scene-0924','scene-0925', 'scene-0926','scene-0927','scene-0928','scene-0929','scene-0930','scene-0931','scene-0962', 'scene-0963','scene-0966','scene-0967','scene-0968','scene-0969','scene-0971','scene-0972', 'scene-1059','scene-1060','scene-1061','scene-1062','scene-1063','scene-1064','scene-1065', 'scene-1066','scene-1067','scene-1068','scene-1069','scene-1070','scene-1071','scene-1072', 'scene-1073'.

A.2 Split based on the geographical-splits[49]

A.2.1 Training Set(40 scenes)

The scenes used for the training set are:

'scene-0563', 'scene-0247', 'scene-0222', 'scene-0792', 'scene-0290', 'scene-0360', 'scene-0075', 'scene-1024', 'scene-0332', 'scene-0638', 'scene-0444', 'scene-0667', 'scene-0914', 'scene-0519', 'scene-0095', 'scene-0045', 'scene-0063', 'scene-0293', 'scene-0872', 'scene-0909', 'scene-0254', 'scene-0672', 'scene-0248', 'scene-0656', 'scene-0525', 'scene-0104', 'scene-0796', 'scene-1053', 'scene-0763', 'scene-0851', 'scene-0138', 'scene-0811', 'scene-0795', 'scene-0696', 'scene-0671', 'scene-0302', 'scene-0639', 'scene-0989', 'scene-0260', 'scene-0653'.

A.2.2 Validation Set(210 scenes)

The scenes used for the validation set are:

'scene-0002', 'scene-0019', 'scene-0043', 'scene-0046', 'scene-0151', 'scene-0158', 'scene-0159', 'scene-0348', 'scene-0355', 'scene-0356', 'scene-0357', 'scene-0358', 'scene-0945', 'scene-0947', 'scene-0981', 'scene-0982', 'scene-0983', 'scene-0018', 'scene-0036', 'scene-0268', 'scene-0275', 'scene-0276', 'scene-0344', 'scene-0345', 'scene-0411', 'scene-1082', 'scene-1083', 'scene-1084', 'scene-1108', 'scene-1109', 'scene-1110', 'scene-0182', 'scene-0183', 'scene-0315', 'scene-0423', 'scene-0424', 'scene-0425', 'scene-0860', 'scene-0861', 'scene-0862', 'scene-0863', 'scene-0864', 'scene-1004', 'scene-1005', 'scene-1006', 'scene-1007', 'scene-0925', 'scene-0926', 'scene-0927', 'scene-0928', 'scene-0071', 'scene-0170', 'scene-0171', 'scene-0172', 'scene-0173', 'scene-0174', 'scene-0175', 'scene-0209', 'scene-0210', 'scene-0211', 'scene-0212', 'scene-0447', 'scene-0448', 'scene-0449', 'scene-0450', 'scene-0500', 'scene-0501', 'scene-0518', 'scene-0599', 'scene-0600', 'scene-0660', 'scene-0661', 'scene-0662', 'scene-0663', 'scene-0664', 'scene-0738', 'scene-0816', 'scene-0821', 'scene-0887', 'scene-0888', 'scene-0889', 'scene-0890', 'scene-0891', 'scene-0892', 'scene-0894', 'scene-0895', 'scene-0896', 'scene-0902', 'scene-0903', 'scene-0109', 'scene-0331', 'scene-0523', 'scene-0627', 'scene-0007', 'scene-0008', 'scene-0009', 'scene-0024', 'scene-

0025', 'scene-0026', 'scene-0027', 'scene-0028', 'scene-0029', 'scene-0030', 'scene-0042', 'scene-0050', 'scene-0057', 'scene-0123', 'scene-0124', 'scene-0154', 'scene-0155', 'scene-0364', 'scene-0365', 'scene-0370', 'scene-0379', 'scene-0380', 'scene-0383', 'scene-0952', 'scene-0953', 'scene-0955', 'scene-0956', 'scene-0957', 'scene-0958', 'scene-0959', 'scene-0960', 'scene-0016', 'scene-0966', 'scene-0413', 'scene-0414', 'scene-0415', 'scene-0416', 'scene-0417', 'scene-1056', 'scene-1057', 'scene-1061', 'scene-1062', 'scene-0184', 'scene-0185', 'scene-0187', 'scene-0188', 'scene-0316', 'scene-0427', 'scene-0428', 'scene-0429', 'scene-0430', 'scene-0858', 'scene-0992', 'scene-1008', 'scene-1009', 'scene-1010', 'scene-1012', 'scene-1013', 'scene-1014', 'scene-1015', 'scene-1025', 'scene-0919', 'scene-0920', 'scene-0921', 'scene-0924', 'scene-0069', 'scene-0073', 'scene-0176', 'scene-0207', 'scene-0208', 'scene-0213', 'scene-0263', 'scene-0396', 'scene-0397', 'scene-0451', 'scene-0452', 'scene-0509', 'scene-0528', 'scene-0529', 'scene-0530', 'scene-0531', 'scene-0532', 'scene-0533', 'scene-0534', 'scene-0535', 'scene-0536', 'scene-0568', 'scene-0570', 'scene-0571', 'scene-0572', 'scene-0573', 'scene-0575', 'scene-0576', 'scene-0650', 'scene-0651', 'scene-0652', 'scene-0658', 'scene-0744', 'scene-0747', 'scene-0749', 'scene-0750', 'scene-0751', 'scene-0752', 'scene-0758', 'scene-0759', 'scene-0760', 'scene-0817', 'scene-0110', 'scene-0330', 'scene-0629', 'scene-0630', 'scene-0632', 'scene-0633', 'scene-0634', 'scene-0635', 'scene-0636', 'scene-0915'.