Multi-Modal Perception for Robots in Business Scenarios

Falong FAN, 221041029@link.cuhk.edu.cn

2025-03-08

Contents

FAN Falong

Professor ZHANG Ruimao [1], Dr. MAO Sitong

Master of Science in Data Science

2025-03-08

Abstract

The first problem we met is how to deal with increasingly appear new objects. We define this problem as class incremental learning problem, and utilize knowledge distillation and sample replay to equip the model with new ability to detect novel objects while still preserve knowledge of old classes. Our method proves that only 3% of old data can keep those old patterns in memory. The second key problem is how to efficiently fuse information capture by different sensors, which are cameras and LiDAR. We represent image features and point cloud features in unified BEV feature space and then use encoder from Transformer to allow them interact with each other.

However, due to limited resources, our methods still has a lot of room for improvement. For example, collecting training images of higher quality can be a feasible way to improve the transferable ability of our model in class incremental learning problem. More complex sensor fusion methods can be tried as well to see how these information can work in a more complementary way.

Key Words: 2D Object Detection, Class Incremental Learning, Knowledge Distillation, 3D Object Detection, Sensor Fusion, BEV Feature Representation, Robots Perception

[1]Contact email: ruimao.zhang@ieee.org

1 Background

The business plan of our group is to provide an online platform for robots management, including monitoring the machines, enabling skills, maintaining the machines relying on cloud computing. During my internship, my mentor Dr. MAO and I were responsible for developing multi-sensor perception skills for the robots. My specific role was to develop object detection algorithms for the robot's skills, including 2D object detection using images and 3D object detection using images and point cloud.

The algorithm solutions we choose for this task is YOLO series [1, 2, 3, 4, 5] for 2D object detection and bird's-eye view (BEV) representation methods such as BEVFusion [6] and BEVFormer [7] using Transformer [8] based structure for 3D object detection, so the report will be focus on development and implementation of these algorithms.

2 Introduction

As a mainstream task in computer vision, objection detection is widely applied in autopilot system in the industry, such as Tesla. For robot automatic obstacle avoidance, vision perception is one of the most advantageous skills as it can see obstacles or objects as what human eyes do.

In our business scenario, there is a probability that the robots will encounter novel obstacles or objects never seen before, so they must be taught to recognize those new objects. In this case, the original model deployed in robots is not able to recognize those objects. Motivated by this scenario, it is required that the robots can learning new patterns continuously while still preserve capabilities to detect original ones.

Typical solutions include training model using a combination of old and new data, or adding a new detection head to the model. Class incremental learning is one of the solutions to this situation, and we introduce 2 training stage to extract new pattern from new data and using knowledge

distillation to keep the model from forgetting old patterns. Specifically, the loss function is guided by two part of labels, with one is the ground truth labels of training data of new classes, another is pseudo labels generated by teacher net. After 2 training stage are completed, it is expected that the student net is able to recognize new classes without triggering catastrophic forgetting [9].

While camera capture rich semantic details, images lack depth information, which is vital for 3D object detection. So LiDARs are introduced to provide spatial information. Since various CNN models are successfully applied to 2D object detection, some researchers have attempted to process RGB-D data using 2D CNN algorithms by projecting LiDAR information extracted from point clouds onto images. However, different depth from point cloud can appear very close in images, this LiDAR-to-camera projection can loss the most part of geometric structure which is carried in original point cloud. Therefore, we are motivated to fuse point cloud and images together to enable LiDAR and camera complement each other, especially representing these features in birds-eye view space.

During my internship, mainly two works were completed. The first task was to equip the robots with ability to continuously learn new patterns, and another one was to enable the robots to understand the surroundings using both images and point cloud from cameras and LiDARs, respectively. Specifically, considering the temporal ability to extract history information in BEVFormer [7] and strong capability of combining modal data in BEVFusion [6], we take advantage of both of the models to fuse point cloud and images while keep the model temporally aware so as to improve the perception skills of the robots.

3 Related Work

2D Object detection is one of the most challenging tasks in the field of computer vision [10]. Basically, the models are asked to output the boxes, coordinates and classes of the interesting objects, sometimes with confidence. Ever since the prevalence of deep neural networks, there are

two main streams of object detection approaches, divided as one stage object detection and two stage object detection. Recently, the overwhelming dominance of Transformer [8] has brought another kind of approach for object detection tasks, such as DETR [11]. For 3D perception tasks, images and point cloud features are usually extracted as bird's-eye view (BEV) representations and specific tasks such as 3D object detection or semantic segmentation are performed in these BEV feature spaces.

Usually, we denote output of 2D object detection results as (x, y, w, h) with center point (x, y) and weight and height (w, h). For 3D bounding boxes, $(x, y, z, w, h, l, \theta, \phi, \psi)$ are denoted with center point (x, y, z) and weight, height, length (w, h, l) and pose (θ, ϕ, ψ).

3.1 2D Object Detection.

Two Stage Object Detection. Namely, there are two stages in this kind of methods. First stage is region proposal network (RPN, [12]) or selective search [10, 13] for proposing some boxes with objects, and the second stage will predict explicit classes, coordinates of the objects. Receiving higher accuracy, these methods sacrifice processing speed. Using selective search to propose 1,000 to 2,000 candidate boxes, R-CNN [10] will extract feature of each box using deep neural network, and then feed the extracted features into support vector machine (SVM, [14]) to implement classification task. However, there may be some overlapped region of extracted feature from candidate boxes, where deep neural network will conduct extraction work repeatedly, resulting in wasting resources. So Fast R-CNN [13] tackled with this problem by extracting features only once and then fitting the selected boxes based on the final feature maps. Region proposal network is introduced in Faster R-CNN [12], which will propose a bunch of boxes using a deep neural network, speeding up the processing speed of the whole network.

Instead of RoI pooling layer, Mask R-CNN [15] uses RoIAlign layer and propose pixel-wise binary prediction for object detection. Considering feature from one scale is not sufficient to capture

detailed information, [16] design feature pyramid network (FPN) to fuse various scale features to improve the performance on small objects. However, for two stage detectors, a pre-defined IoU threshold is a must for propose foreground boxes. The higher the threshold, the higher quality of the proposed boxes. But higher threshold could lead to less number of training data, enlarge the chance the model can overfit. Therefore, [17] designed a cascade structure, namely Cascade R-CNN, to gradually increase the IoU threshold while still keep a considerable number of positive boxes. To enlarge receptive field, TridentNet [18] uses trident block with dilated convolution [19] to detect difference scale objects, and sets these multi-branches to be parallel to faster the forward speed.

One Stage 2D Object Detection. As a classical one stage algorithm, Single shot multibox detector (SSD, [20]), using feature pyramid structure to detect objects. Specifically, the model will output various scale of detection from different layers without RPN. Another prevalent one stage algorithm is YOLO series [1, 2, 3, 4]. Simply cutting the feature maps into grids, YOLO series [1, 2, 3, 4] can detect objects faster than two stage ones, receiving real-time detection. YOLOv3 [3] combines feature pyramid mechanism and Res unit to detect smaller objects and allowing the network to become deeper. YOLOv4 [4] and YOLOv5 [5] are slighter to deploy in terminal such as cell phones and robots. Thus, YOLO series [1, 2, 3, 4] are more frequently employed in industry and can perform real-time object detection in a more satisfactory way.

While some researchers allocate more efforts on the structure of the detectors, [21] proposed focal loss to deal with hard samples–small objects usually, and proved the effect of this loss on well-designed one stage detector RetinaNet. Some works also devote efforts on the efficiency of the model, such as EfficientDet [22] introduces BiFPN and take advantage of EfficientNet [23] to faster the inference speed.

Transformer-based 2D Object Detection. Transformer [8] is famous for encoder-decoder mechanism in natural language processing, attracting plenty of researchers devoting efforts to

transfer this structure to other fields. Previoysly, no matter for one or two stage algorithms, the models are working with manual intervention, mainly reflecting in RPN [13], setting size and ratio of anchors and non-max-suppression (NMS) and so on. There exist some work trying to replace some of the layers of CNN structure with self-attention mechanism, but these work are still under manual supervision such as NMS.

DEtectional TRansformer (DETR, [24]) are different from these work as the model require no prior knowledge, achieving end-to-end object detection. To be specific, DETR [24] do not rely on pre-defined anchors or non-max-suppression operation. Other researchers have attempted to improve the performance of DETR. Inspired by [25] and the success of multi-scale feature in one or two stage object detection, [11] designed multi-scale deformable attention mechanism to sample several interesting point instead of making attention within whole feature map, fastering the inference speed and improving the accuracy of small objects of DETR-based methods. By designing dynamic heads, [26] make the attention mechanism be aware of scale, spatial and task features, respectively. [27] sets queries as boxes coordinates in cross-attention in decoder, and update these positional information in each decoder layer, speeding up the training convergence of DETR. Noise are added to class embedding of cross-attention in decoder to enable the model to denoise, which will speed up Hungarian matching process in [28].

3.2 Class-Incremental Learning and Knowledge Distillation

The object of class-incremental learning is to learn new patterns while the model still preserve original capabilities to recognize old classes. Mathematically, at incremental stage t, we have a new training dataset $\mathcal{D}^t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$, where $\mathbf{x} \in \mathcal{X}$ is the samples which model never seen before from sample space $\mathcal{X}$ and $\mathbf{y} \in \mathcal{C}_t$ is the new labels. Specifically, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ if $i \neq j$, meaning that the classes in incremental stage t have never appeared in previous stage t', where $0 < t' < t$.

But class incremental learning always faces with catastrophic forgetting [29, 30], i.e., the model tends to forget about old patterns while learning the new ones. There are three approaches to alleviate this problem, one is to modified the model structure, one is to replay-based, such as samples or features, when learning new knowledge [31]. And last but not least, some research will punish some vital parameters of the model to avoid the model move to far from original solution space. For example, by applying Frisher information matrix to estimate the importance of model parameters, [32] introduce elastic weight consolidation (EWC) to constrain the sharp change of the model weights. [33] used memory aware synapses (MAS) to measure the sensitivity of model parameters to changes. But some researches such as [34, 35] proved that regularization and punishments on model parameters sometimes can perform worse under class incremental setting.

One typical way to reconstruct the model is based on loss function, which will impose constraint on gradient direction to protect old knowledge from being cover by new knowledge without accessing to old data. Hinton *et al.* [36] proposed knowledge distillation to guide a smaller net to learning patterns from a larger model with higher accuracy and better performance, named teacher model. [37] introduced dual augmentation, namely class augmentation and semantic augmentation, to encourage the model to learn more diverse features so it will not be confused by new distribution of feature when it comes to new patterns. While cross-entropy loss is used to calculate novel classes, Li *et al.*[38] take advantage of knowledge distillation mechanism to preserve old patterns. Hao *et al.* [39] proposed Class-Incremental Faster R-CNN (CIFRCN) model to learning new classes with only new data. Typically, they expanded RPN and applied distill knowledge of old patterns from classifiers.

Another way to maintain old performance is to train the model with replayed data. [40] firstly introduced iCaRL using heuristic sampler, which is Herding [41], and only need to keep a small number of training sample per class. While [40] retain the most representative samples, [42] consider the samples on the class boundary should be kept so as to obtain a clearer boundary for inter-class classification. [40] use knowledge distillation to keep original performance, but goes

further than [38] by accessing to old data while training the model to learning new knowledge, improving the performance of model on original classes. Samples replay method is explored with samples statistic as well, sometimes even without any samples. For example, in [43], they replayed previous samples as well as applied sample statistics, which they call dual memory to utilize more prior information about old dataset. Instead preserving old samples, [44] proposed that replaying features will be more memory-efficient. For example, saving a 512-dimension feature other than a $256 \times 256 \times 3$ image takes an order of magnitude less memory. But old features space maybe not consistent with new feature space while new features are extracted from iteratively-updated model. Thus, [44] also designed a feature adaptation network (FAN) to project old features into new feature space to improve the consistency between the old and new one. Other works also attempt to preserve activated values in hidden layers [45] or the encoded tensor of the hidden layers [46].

To tackle with class incremental learning problem, our method applies knowledge distillation with sample-replaying to keep old knowledge. Specifically, different from [40], in which they replayed old data using Herding selecting approach [41], we firstly shuffle the whole old dataset, and then only replay 3% of those samples randomly.

3.3 3D Object Detection.

Camera-based 3D Object Detection. Inspired by 2D object detection models, researchers such as Brazil et. al [47] and [48, 49, 50, 51] predict 3D objects based on 2D predictions. More recently, [52] introduced FCOS3D to discard pre-procedure of 2D detectors to directly output 3D bounding boxes. But taking into account the geometric information of the object, [53] construct geometric relation graphs and probabilistic representation to estimation the depth information instead estimating depth with the fact that regarding instances as isolated [49, 47, 54]. Using dynamic voxelization to remove pre-defined points number for each voxel, [55] proposed multi-view fusion to enable the points to capture latent information from birds-eye view and perspective view.

DETR [24] introduce a novel stream for object detection without any post-processing, which encourages [56] to design learnable queries of an end-to-end detector called DETR3D [56] for 3D object recognition. Instead of perspective view, the idea of BEV representation from LiDAR-based models innovate another kind of methods. For example, [57] introduced categorical depth distribution to project image features into BEV space, while [58] using pseudo-lidar to extract images into BEV feature representation by estimating depth information. BEVFormer [7] combine current BEV queries with history BEV queries to extract the temporal information for 3D perceptions.

LiDAR-based 3D Object Detection. Different from camera-based 3D object detection, LiDAR-based detectors usually express features in BEV representation or voxel representation. PointNet [59] and PointNet++ [60] firstly extract point cloud features and then represent them in BEV space, which are then used to detect objects. [61, 62] introduce two stage models, one for point cloud segmentation to propose potential bounding boxes, and the another is used to refine the candidate boxes.

To efficiently conduct feature engineering, [63] propose VoxelNet, in which voxel representation is firstly introduced, to convert point cloud into voxel, extracting point cloud information by designing voxel feature encoder into unified feature representation. But calculating on voxel still time-consuming. Inspired by sparse convolutional neural networks (SCNN) [64], [65, 66] faster the forward speed using sparse kernel. The method we use for point cloud feature engineering is SECOND [66]. Considering the efficient computation of voxel-based networks and large receptive fields of PointNet-baesd networks, [67, 68] integrate these two models to largely improve the detection skills on point cloud. LiDAR-RCNN [69] attempts to make the model to be aware of the size of proposal. All these works are two stage 3D object detection with RCNN framework. Pointpillars [70] firstly encodes point cloud into pillars, and then extracts features from these stacked pillars to generate pseudo images, which will be fed into SSD [20] to conduct predictions.

Multi-Modal 3D Object Detection. Point cloud preserve more depth perception information while RGB images are more semantically aware. MV3D [71] extract point cloud features, front-view features and images features respectively before these features are fused. Frustum PointNet [72] and Frustum ConvNet [73] generate 3D frustum based on 2D proposals, and point cloud are utilized to estimate the depth of the 3D bounding boxes. FUTR3D [74] defines backbone for each modal to encode the features respectively, and propose 3D object queries to predict 3D bounding boxes in a unified Transformer decoder based on the fusion features, while TransFusion [75] has prediction decoder for each modal data. After extracted from corresponding backbone in BEVFusion [6], point cloud features and image features are fused in BEV representation space, following a Transformer to encode and decode these feature before they are fed into prediction head.

4 Model

4.1 Model for Class Incremental Learning

4.1.1 Model Structure

Main idea of our class incremental learning solution using knowledge distillation involves *two* training stages. The backbone of the model, identical with and initialized using YOLOv5s6 [5], is frozen in training process.

In pretraining stage, we concatenate a semantic segmentation head to the backbone together with a classification head to train the model using old classification dataset and segmentation dataset. In the *first* training stage, a brand new head is concatenated to the backbone as shown in the blue part of Fig. 2. This new-class head and its corresponding neck are trainable in training stage 1, and is trained merely by new object dataset. In the *second* training stage, another brand new head is added to the model whose neck is also learnable. In this stage, the model is trained under two types of labels: pseudo-labels generated by the output of the old-class head and the new-class head, and

ground truth labels. This allows the model to distill all the old and new patterns of the dataset into one head, namely the oldNnew-class head, without bias towards either the old or new patterns. The idea of these two training stage is similar with linear probe protocol but with more learnable layers. In inference stage, only the oldNnew-class head is applied to predict objects, as shown in Fig. 2, which is identical with the pretrained model.

Pseudo code in Algorithm 1 shows the whole training process of our methods. Without further noting, $\mathcal{F}_{seg}$, $\mathcal{F}_{old}$ and $\mathcal{F}_{oldNnew}$ share the same backbone os as the parameters of the backbone.

Algorithm 1: Two Stage Training Process

1 Pretrain:
Data: segmentation data $\mathcal{D}_{seg}$, old class data $\mathcal{D}_{old}$
Result: Segmentation prediction: $\mathcal{F}_{seg}(\mathcal{D}_{seg};\theta_0)$, Object detection result: $\mathcal{F}_{old}(\mathcal{D}_{old};\theta_0)$
2 Training Stage 1:
Data: new class data $\mathcal{D}_{new}$
Result: Object detection result: $\mathcal{F}_{new}(\mathcal{D}_{new};\theta_1)$
3 Training Stage 2:
Data: new class data $\mathcal{D}_{new}$, 3% old class data $\mathcal{D}_{old}$
Result: Object detection result: $\mathcal{F}_{oldNnew}(\mathcal{D}_{new};\theta_2)$
4 Inference:
Data: any inference data $\mathcal{D}$
Result: $\mathcal{F}_{seg}(\mathcal{D};\theta_0), \mathcal{F}_{oldNnew}(\mathcal{D};\theta_2)$

4.1.2 Loss Function

Loss function for YOLOv5 [5] consists of three parts, including bounding box regression loss $\mathcal{L}_{bbox}$ using CIoU [76], classification loss $\mathcal{L}_{cls}$ using Focal loss [21] and confidence loss $\mathcal{L}_{conf}$ using binary cross entropy loss as Eq. 1 with $a = 0.5, b = 0.05$ and $c = 1.0$ being importance factor of each loss.

$$\mathcal{L}(X,\theta_1) = a \cdot \mathcal{L}_{cls}(X,\theta_1) + b \cdot \mathcal{L}_{bbox}(X,\theta_1) + c \cdot \mathcal{L}_{conf}(X,\theta_1) \tag{1}$$

Inspired by CIoU loss in [76], bounding box regression loss considering Intersection over Union (IoU) between predicted bounding box and ground truth box, overlapped area, distance between

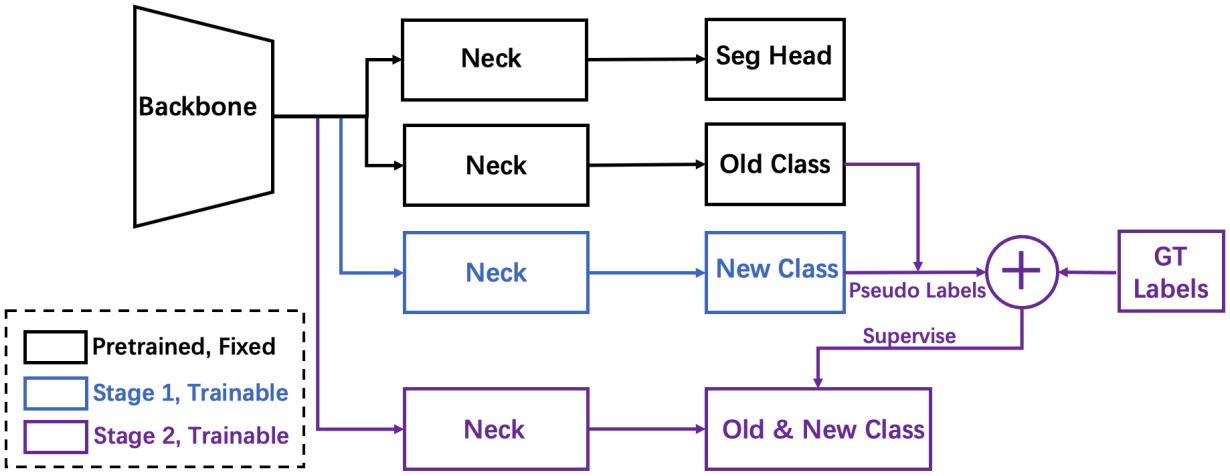


Figure 1: Model in Training Stage. The pre-trained model can conduct segmentation tasks and detect old class as shown in black boxes. In the first training stage, the backbone is frozen, while neck and detect heads are learnable. Blue boxes are trained in training stage 1, while the purple ones are trained in training stage 2. Outputs from old-class and new-class head are used as pseudo labels to guide the learning process of oldNnew-class head in training stage, together with the ground truth labels.

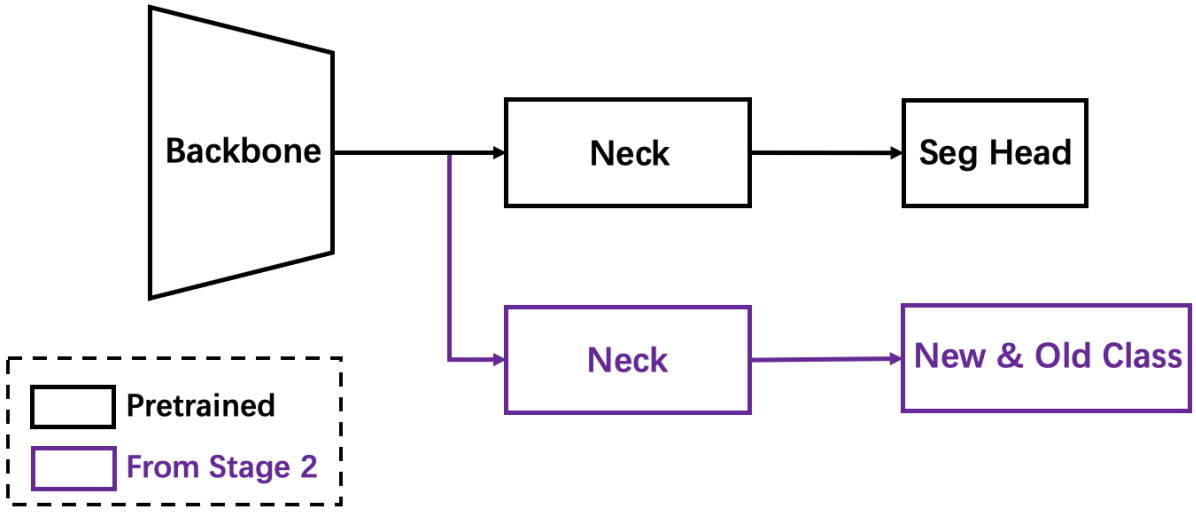


Figure 2: Model in inference stage. Only head from training stage 2 is used to implement object detection, together with segmentation head from pre-trained model.

the center of each box, and the ratio of width and height. With ground truth box denoting as $B_{gt} = (x_{gt}, y_{gt}, w_{gt}, h_{gt})$ and predicted bounding box as $B_p = (x_p, y_p, w_p, h_p)$, IoU can be calculated as Eq. 2.

$$IoU = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} \tag{2}$$

Then, bounding box regression loss is calculated as Eq. 3 using CIoU,

$$\mathcal{L}_{bbox}(X, \theta_1) = 1 - CIoU \tag{3}$$

$$where\, CIoU = IoU - \frac{\rho^2}{c^2} - \alpha v \tag{4}$$

where $v = \frac{4}{\pi^2}(\arctan\frac{w_{gt}}{h_{gt}} - \arctan\frac{w_p}{h_p})^2$ reflects the ratio of width and height, c denotes the diagonal length of the smallest enclosing box covering ground truth box and bounding box, and $\rho = \sqrt{(x_p - x_{gt})^2 + (y_p - y_{gt})^2}$ represents the center point distance between B_p and B_{gt}, and α is the coefficient of v. Basically, bounding box loss as Eq. 3 will guide the model to output bigger overlapped area, closer center distance of ground truth box and predicted one, smaller enclosed rectangular and more similar ratio of width and height.

Binary cross entropy loss is applied to measure the confidence of objects as Eq. 5,

$$\mathcal{L}_{conf}(X, \theta_1) = -L_{conf} \cdot \log C(X, \theta_1) - (1 - L_{conf} \cdot \log(1 - C(X, \theta_1))) \tag{5}$$

where L_{conf}, implying the ground truth confidence, is identical with CIoU if corresponding value greater than 0, otherwise 0, and $C(X, \theta_1)$ is the predicted confidence matrix. Basically, if there exists interested objects in bounding box, we want the model to be more confident in predicting this fact.

Small objects, or objects that have small number in the whole dataset, are usually difficult to be detected. Focal loss introduced by Lin et. al [21] is good at dealing with these hard samples as

shown in Eq. 6. Thus, focal is employed to power the model to pay more attention to small objects recognition in YOLOv5 [5].

$$\mathcal{L}_{cls} = -(L_{gt})^{\gamma} \cdot \log P(X, \theta_1) - (1 - L_{gt})^{\gamma} \cdot \log(1 - P(X, \theta_1)) \tag{6}$$

where L_{cls} represents the one-hot ground truth labels and $P(X, \theta_1)$ is the predicted logits. $\gamma \in [0, 5]$ is the focusing parameter, and is setting as 1.5. The greater the focusing parameter, the more attention will the model pay to the difficult samples.

4.2 Model for Multi Sensors 3D Object Detection

4.2.1 Model Structure

BEVFusion [6] takes images and point cloud as input and extract features using Transformer into consistent BEV space. Using the spatial and geometric information fused from image features and point cloud features respectively, the model can perform 3D object detection and semantic segmentation tasks simultaneously as shown in the top half in Fig. 3. Instead of multi sensors, BEVFormer [7] takes only images from multi views as input and enable these images interact between each other with the help of spatial cross-attention structure. Historic information from temporal self-attention also facilitate the model to better understand the surrounding environment.

It turns out that the way how point cloud information is extracted in BEVFusion can be effectively fused with images information, and BEVFormer structure has a better interaction between different views and history queries. Therefore, we attempt to aggregate the pros of these two models together to generate a novel model that is not only able to interact between views and history information but also able to fuse perception information from camera and LiDAR.

Specifically, point cloud features are extracted using a backbone such as SECOND [66] and PointPillars [70], while the images are fed into traditional convolutional network to generate fea-

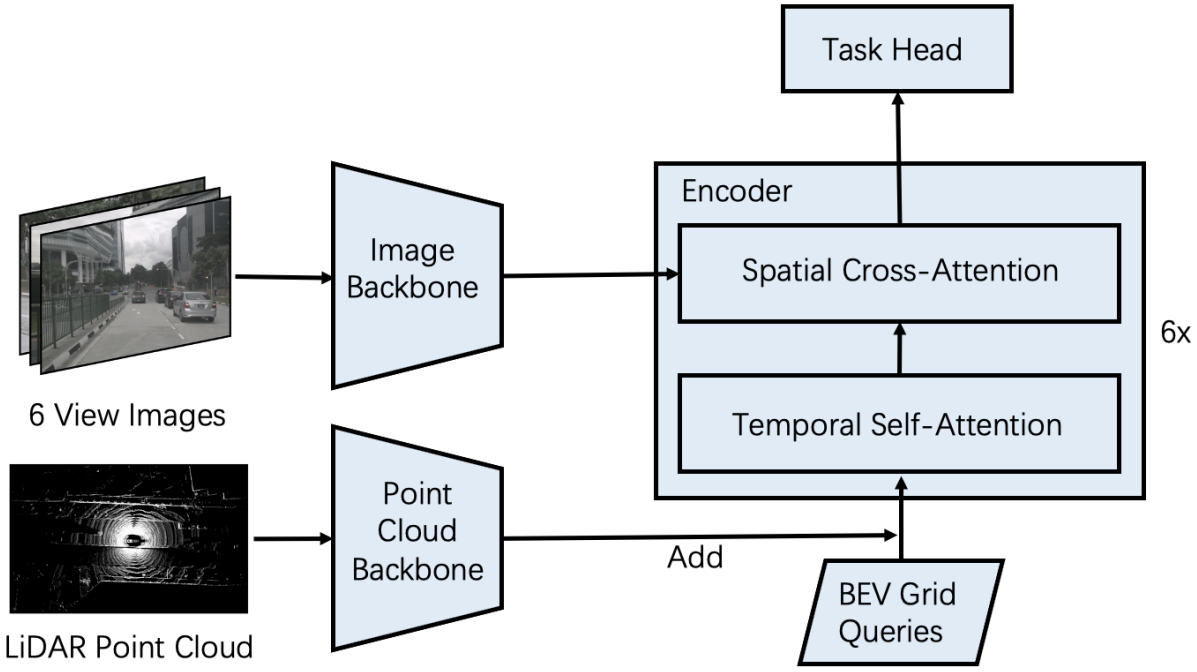


Figure 3: **Model for multi sensors 3D object detection.** Point cloud feature and image features are extracted from corresponding backbone. Since BEV grid queries is initialized, point cloud features are added to BEV queries. The concatenating result is fed into encoder to make interaction with 6-view image features. Finally, task-specific head will take output from decoder to conduct predictions.

ture representations. ResNet-101 [77] are chosen in this case. After initializing BEV queries, point cloud features are added to these queries. Temporal self-attention will interact between history and current BEV queries, with spatial cross-attention following to extract key information from image representations and point cloud feature. After object queries, which carry massive information about the surroundings around the robots captured by cameras and LiDAR, is generated from encoder, task-awared prediction head will take these object queries as input to perform prediction task. In our experiment, the task is to predict 3D bounding box of objects.

4.2.2 Loss Function

Objective function for training a 3D detection using multi-modal model consists of two parts, including classification loss $\mathcal{L}_{cls}$ and bounding box regression loss $\mathcal{L}_{bbox}$ as Eq. 7:

$$\mathcal{L}(X;\theta) = a \cdot \mathcal{L}_{cls}(X;\theta) + b \cdot \mathcal{L}_{bbox}(X;\theta) \tag{7}$$

where $a = 2.0$ and $b = 0.25$. Specifically, $\mathcal{L}_{cls}$ is Focal loss [21], which is introduced in Eq. 6, and L1 loss is applied to compute 3D bounding box regression loss. X is input and θ is the parameters of model.

Python style pseudo codes are provided as the following code block. In our pseudo code, we assume the shape of input images is `[W, H, C]` and `[N, d]` for point cloud. BEV grid is set as `[bev_h, bev_w]` with dimension of queries being `embed_dims`. Basically, forward procedure can be divided into 4 parts. Firstly, features of different sensors are extracted from corresponding backbones. Then BEV queries is initialized and added with point cloud features. Following is the information interaction between images and point cloud features. Inside the encoder, there are temporal self-attention between history and current BEV queries and spatial cross-attention between images and point cloud. Finally, after predicting, loss is calculated for backward propagation.

```
# input          - X_img, [W, H, C],
# input          - X_pc, [N, d]
# encoder        - encoder as BEVFormer
# backbone_img   - backbone of images, ResNet-101
# backbone_pc    - backbone of point cloud, SECOND or PointPillars
# head           - task aware prediction head

# Feature Extraction
feat_img = backbone_img(X_img)                # bev_h * bev_w * embed_dims
feat_pc = backbone_pc(X_pc)                   # bev_h * bev_w * embed_dims

# Feature Aggregation
bev_queries = nn.Embedding(bev_h*bev_w, embed_dims)
bev_queries = bev_queries + feat_img          # bev_h * bev_w * embed_dims

# Encode feature from images and point cloud
query = encoder(bev_queries, query_img)

# Prediction
prediction = head(query)

# Loss
loss = 0.75 * GaussianFocalLoss(prediction, GT) + \
       0.25 * SmoothL1Loss(prediction, GT)
```

5 Experiment

5.1 Experiments on Class Incremental Learning

In the experiment for class incremental learning, we attempt to equip the model with ability to detect lampposts without reducing performance on old tasks sharply. In this case, we prepared 340

real-world lamppost images, and 80 classes from COCO [78] are served as old objects.

Figure 4: **Inference results.** 1^{st} column images are the head in pre-trained model, 2^{nd} column images are the head in training stage 1, 3^{rd} column images are the head in training stage 2. Person and lamppost can be detected respectively in the 1^{st} and 2^{nd} column sub-figures, reflecting the ability old and new head. And the 3^{rd} column sub-figures successfully detect both person and lamppost, showing old and new patterns are learned by single on head.

5.1.1 Pre-trained Model

Table 1: Experiment results of class incremental learning. The following results only show object detection task without segmentation task.

Training Stage	Dataset	Epoch	mAP
Pre-trained	COCO	50	0.540
Training Stage 1	Lamppost	50	0.44
Training Stage 2	3% COCO + Lamppost	50	0.520

As shown in Fig. 2, the pre-trained model consists of two task heads, one is semantic segmentation head and another is old-class head. The backbone of the whole model, which is Darknet-53, is sharable. Pre-trained model is trained using semantic segmentation dataset and 2D object detection dataset. While detection task is trained using COCO dataset [78], semantic segmentation dataset is confidential data in our group, so it will not be discussed in details. After 50 epochs of training is finished on RTX 5000 with 16 GB, pre-trained model is well prepared with mAP being 0.540 on

old classes. As shown in the first sub-figure in Fig. 4, the model can detect the person showing in the scene. Detailed overview table shows in Table 1.

5.1.2 Training Procedure

Training Stage 1. In the first training stage, the backbone, heads and necks of segmentation task and old class are set frozen while the head and neck of new class are learnable, and 50 epochs are set for those modules to learn new pattern. As shown in the second sub-figure in Fig. 4, new task head is able to detect lamppost successfully with mAP being 0.44 as in Table 1.

Training Stage 2. In the second training stage, brand new neck and head for detecting both old and new classes are added after backbone of the model. We call this branch as oldNnew head. Being the same as training stage 1, parameters of the new neck and head are required to learn knowledge of old and new object, and the other modules are frozen. Specifically, 3% of COCO dataset [78] are randomly sampled and all the new lamppost images are fed into the model, and outputs from both new class and old class serve as pseudo labels to encourage the oldNnew head distill patterns. In detail, if the training image is from COCO [78], new class head will output possible signs of new class, and add these signs to the original COCO [78] labels. Procedure works the same when the images are from lamppost, correspondingly.

When pseudo labels are aligning with ground truth labels, the model make it to detect both old and new classes with mAP being 0.520 after 50 epochs as illustrate in Table 1. In the last figure in Fig. 4, both the person and the lamppost are quite perfectly bounding.

5.1.3 Inference Stage

After finishing 2 training stage, only heads and necks of segmentation task and training stage 2 are kept. Therefore, when inferencing, oldNnew head can output all the classes as shown in the third picture in Fig. 4. In this case, we have a model with identical structure of pre-trained model, but equip the new model with new detection ability.

5.2 Experiments on Multi-Modal Perception

The nuScenes dataset [79] with 1.4 million 3D annotations for 10 categories are used to trained the multi-modal perception models. After 24 epochs of training on four NVIDIA A100 with 80 GB for 5 days on nuScenes training dataset, mAP performance of our model is outperforming the original BEVFormer [7] by 3.8%, reaching 48.3% on validation dataset. In these experiments, we set batch size to be 4 and the other hyper-parameters are the same with experiment setting in BEVFormer [7].

Table 2: 3D object detection results on nuScenes [79] `test` set. "C" and "L" indicate camera and LiDAR respectively.

Method	Modality	Backbone for LiDAR	mAP
BEVFormer [7]	C	-	0.445
Ours	C + L	SECOND [66]	**0.478**
Ours	C + L	Pointpillars [70]	**0.483**

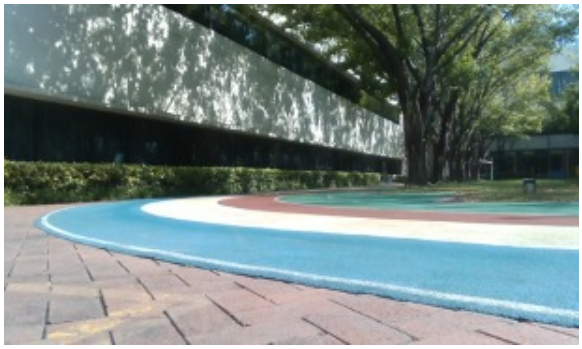
The experiment results are shown in Table 2. Since BEVFormer only takes images as input, which is set as "C" in Table 2, there is no backbone for LiDAR in this case n our experiments. And we have tested two backbone for point cloud, namely SECOND [66] and Pointpillars [70]. It turns out that complement from spatial information which is extracted from point cloud can improve the performance of BEVFormer [7].

6 Limitations

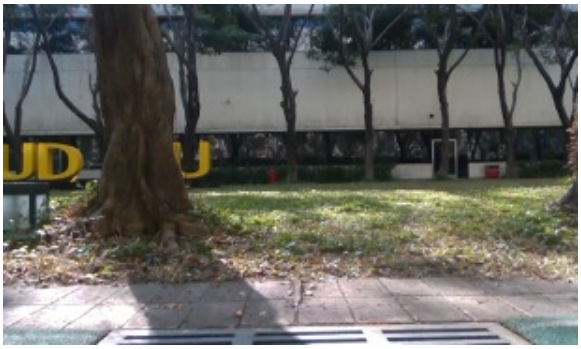
The experiment results for class incremental learning shows that the model can detect lamppost successfully without trigerring heavy catastrophic forgetting, illustrating that knowledge distillation can efficiently deal with class incremental problem with the assistance of a very small proportion of old dataset. But since the limitation to the expenditure on manpower and resources, lamppost dataset is collected by the camera on the robots, resulting in various quality of these images. For example, in Fig. 5(a), lamppost is too small to be noticed even for a human being.

Due to intrinsic or external parameters setting, images may be overexposed as Fig. 5(c) or tilted as Fig. 5(d), resulting in the training data out of distribution. Incomplete of the object can also cause difficulties for training the model.

These poorly collected training images result in bad transferable ability of the model. Though lamppost can be detected in inference stage, the model is not very confident at outputing these lamppost predictions. There exists great room for improving the performance of the model if allocating more efforts on collecting and filtering data.

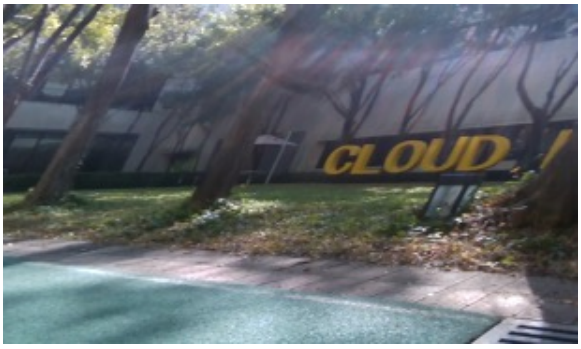


(a) Lamppost 1: Too small.



(b) Lamppost 2: Part of object.

(c) Lamppost 3: Overexposed.



(d) Lamppost 4: Tilted image.

Figure 5: Examples of low quality lamppost images collected by camera installed on robot.

For multi-modal perception, it turns out that when enabling the BEVFormer [7] model to fuse information from images, it can perform better under spatial cross-attention and temporal self-attention with detailed semantic information captured by cameras. However, the way we combine

features from images and point cloud is quite naive. More sophisticated approaches on this operation can be explored to improve the ability of the sensor fusion model.

In addition to supervised learning, unsupervised methods are receiving more and more attention. Since self-supervised learning achieved great success in natural language processing such as GPT series [80, 81, 82], BERT [83] and computer vision such as MAE [84], some researchers are allocating efforts to convert masked self-supervised architectures to point cloud backbone. For example, Voxel-MAE [85] firstly transfer point cloud into voxel, and then conduct binary prediction on voxel using self-generated labels. Occupancy-MAE [86] predict point cloud directly which are randomly masked. We believe that there exists great opportunities to train point cloud backbone without manual labels to enhance our technological business solution.

7 Ablation Study

Before all the solutions finally work, there are several solutions that failed to meet our requirement to tackle with the practical problems we meet. This section will introduce how those abandoned solutions are designed and the reasons they failed.

7.1 Class Incremental Learning

Data Collection. When we are collecting data for lampposts, the first solution we are trying is to generate fake image using a simulated world model built from point cloud. However, due to technical limitations, there is a big gap between the simulated world and the real world. As a result, after training with artificial images, the model cannot transfer to the real world, losing accuracy in detecting objects in the real world, such as humans and car. For artificial images, Fig. 6(a) is the generated artificial image, and Fig. 6(b) is the labels of each pixel which are used for segmentation task.

(a) Artificial image (b) Pixel-wise labels

Figure 6: Images and labels generated by simulated world.

Knowledge Distillation. Two training stages setting with knowledge distillation outperform two previous solutions. The first one simply add new data to the original dataset to form a new training dataset to train a new model. In this case, new data only have annotations of the new classes. However, no matter we trained using common linear probe protocol or from scratch, when the model predicting, old classes and new ones cannot be detected simultaneously. The reason for these phenomena is that the training data never label the old and new objects in one image. Therefore, the model learned a bias that the old objects and new objects can never appear at the same time. Otherwise, the model is confused and have no confidence on output both new and old classes.

Data Replay. In the second solution, we use a teacher model to output the old classes in the new training images, and the model is trained using new dataset merely. But in this situation, performance on old classes became extremely bad since old classes appear rarely in the new dataset, leading to seldom appearance in training procedure. This is how catastrophic forgeting is triggered. Finally, we form a new dataset using 3% old data and all the new images. Specifically, two training stages are set with knowledge distillation applied to distill old patterns from teacher net as introduced previously.

7.2 Sensor Fusion for 3D Object Detection

Multi-modal Fusion. The first solution designed for sensor fusion is to regard point cloud feature as the seventh view in BEVFormer [7], allowing the point cloud features to interact with multi-view images. To be specific, inspired by BEVFusion [6], we firstly extract image and point cloud feature respectively, and then concatenate these feature together as the input of encoder. But this solution did not work as we expected. We analysed that since point cloud features are represented in 3D space while images ones are in BEV space, features from different sensors can not be represented in a unified space, or leading to low inefficient fusion. These 3D space and BEV space obstructed the model sample features of the same objects. Therefore, this solution failed to fuse information from various sensors.

Another straightforward fusion method has been tried as well, that is we consider to allow the point cloud representation to have a shared prediction head with images features. However, this situation did not utilize the advantage of Transformer, forbidden all the interaction between point cloud and images. Finally, we came out with the solution that firstly extract point cloud features and add them to the output of encoder, leaving the images representation to have a better interaction with point cloud in decoder, and this works well.

Acknowledgements

I am deeply grateful for the invaluable assistance and support provided by Prof. ZHANG Ruimao and Dr. MAO Sitong, without whom this work could not have been completed. Their dedication, expertise, and insightful feedback have greatly enhanced the quality of this research.

In addition, I would like to extend my sincere thanks to our group for providing me with an excellent platform, including access to the computational power of 8 NVIDIA A100 with 80 GB each. These infrastructures have played a crucial role in enabling me to carry out the necessary experiments and simulations for this research.

I would also like to express my heartfelt appreciation to my parents for their unwavering support and encouragement throughout my academic journey. Their love, understanding, and encouragement have been the driving force behind my pursuit of excellence, and I am deeply grateful for all that they have done and continue to do.

Finally, I would like to acknowledge the contributions of all those who have assisted me in this research, including my colleagues, friends and classmates. Their guidance and insights have been instrumental in shaping my academic research, and I am deeply grateful for their support.

Works Cited

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[2] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[5] G. Jocher, "Yolov5." `https://github.com/ultralytics/yolov5`, 2020.

[6] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," *arXiv preprint arXiv:2205.13542*, 2022.

[7] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, pp. 1–18, Springer, 2022.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions.," *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[10] 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*, pp. 580–587, 2014.

[11] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[12] 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.

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

[14] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[16] 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*, pp. 2117–2125, 2017.

[17] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018.

[18] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6054–6063, 2019.

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[22] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[23] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[24] 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*, pp. 213–229, Springer, 2020.

[25] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

[26] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang, "Dynamic head: Unifying object detection heads with attentions," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7373–7382, 2021.

[27] S. Liu, F. Li, H. Zhang, X. Yang, X. Qi, H. Su, J. Zhu, and L. Zhang, "Dab-detr: Dynamic anchor boxes are better queries for detr," *arXiv preprint arXiv:2201.12329*, 2022.

[28] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "Dn-detr: Accelerate detr training by introducing query denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13619–13627, 2022.

[29] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[30] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[31] E. Belouadah, A. Popescu, and I. Kanellos, "A comprehensive study of class incremental learning algorithms for visual tasks," *Neural Networks*, vol. 135, pp. 38–54, 2021.

[32] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[33] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

[34] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *arXiv preprint arXiv:1810.12488*, 2018.

[35] G. M. Van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019.

[36] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[37] F. Zhu, Z. Cheng, X.-Y. Zhang, and C.-l. Liu, "Class-incremental learning via dual augmentation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14306–14318, 2021.

[38] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[39] Y. Hao, Y. Fu, Y.-G. Jiang, and Q. Tian, "An end-to-end architecture for class-incremental object detection with knowledge distillation," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2019.

[40] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

[41] M. Welling, "Herding dynamical weights to learn," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1121–1128, 2009.

[42] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

[43] E. Belouadah and A. Popescu, "Il2m: Class incremental learning with dual memory," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 583–592, 2019.

[44] A. Iscen, J. Zhang, S. Lazebnik, and C. Schmid, "Memory-efficient incremental learning through feature adaptation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pp. 699–715, Springer, 2020.

[45] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10203–10209, IEEE, 2020.

[46] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pp. 466–483, Springer, 2020.

[47] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9287–9296, 2019.

[48] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7074–7082, 2017.

[49] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1991–1999, 2019.

[50] B. Xu and Z. Chen, "Multi-level fusion based 3d object detection from monocular images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2345–2353, 2018.

[51] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[52] T. Wang, X. Zhu, J. Pang, and D. Lin, "Fcos3d: Fully convolutional one-stage monocular 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 913–922, 2021.

[53] T. Wang, Z. Xinge, J. Pang, and D. Lin, "Probabilistic and geometric depth: Detecting objects in perspective," in *Conference on Robot Learning*, pp. 1475–1485, PMLR, 2022.

[54] P. Li, H. Zhao, P. Liu, and F. Cao, "Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 644–660, Springer, 2020.

[55] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," in *Conference on Robot Learning*, pp. 923–932, PMLR, 2020.

[56] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," in *Conference on Robot Learning*, pp. 180–191, PMLR, 2022.

[57] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander, "Categorical depth distribution network for monocular 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8555–8564, 2021.

[58] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8445–8453, 2019.

[59] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[60] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[61] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 770–779, 2019.

[62] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 8, pp. 2647–2664, 2020.

[63] 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*, pp. 4490–4499, 2018.

[64] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 806–814, 2015.

[65] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9224–9232, 2018.

[66] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[67] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10529–10538, 2020.

[68] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection," *International Journal of Computer Vision*, pp. 1–21, 2022.

[69] Z. Li, F. Wang, and N. Wang, "Lidar r-cnn: An efficient and universal 3d object detector," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7546–7555, 2021.

[70] 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*, pp. 12697–12705, 2019.

[71] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.

[72] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 918–927, 2018.

[73] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1742–1749, IEEE, 2019.

[74] X. Chen, T. Zhang, Y. Wang, Y. Wang, and H. Zhao, "Futr3d: A unified sensor fusion framework for 3d detection," *arXiv preprint arXiv:2203.10642*, 2022.

[75] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1090–1099, 2022.

[76] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 12993–13000, 2020.

[77] 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*, pp. 770–778, 2016.

[78] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[79] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

[80] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[81] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[82] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[83] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[84] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

[85] C. Min, D. Zhao, L. Xiao, Y. Nie, and B. Dai, "Voxel-mae: Masked autoencoders for pre-training large-scale point clouds," *arXiv preprint arXiv:2206.09900*, 2022.

[86] G. Hess, J. Jaxing, E. Svensson, D. Hagerman, C. Petersson, and L. Svensson, "Masked autoencoders for self-supervised learning on automotive point clouds," *arXiv preprint arXiv:2207.00531*, 2022.

A Appendix

A.1 BEVFormer

As shown in Fig. 7, BEVFormer takes advantage of encoder from Transformer [8], to complete two kinds of interactions, which are temporal interaction and multi-modal interaction. From the model structure, it can be noticed that BEVFormer [7] only takes images as input.

Firstly, history BEV queries and current BEV queries will communicate with each other in temporal self-attention module to extract more accurate information of the same object. In this stage, the model will utilize the camera intrinsic and extrinsic parameters to convert these two BEV queries into a unified coordinate. Then in spatial cross-attention module, features from different views will interact with BEV queriess, sampling points which is interested by BEV queries. After 6 times of encoding procedure, detection and segmentation heads will conduct corresponding predictions based on BEV queries.

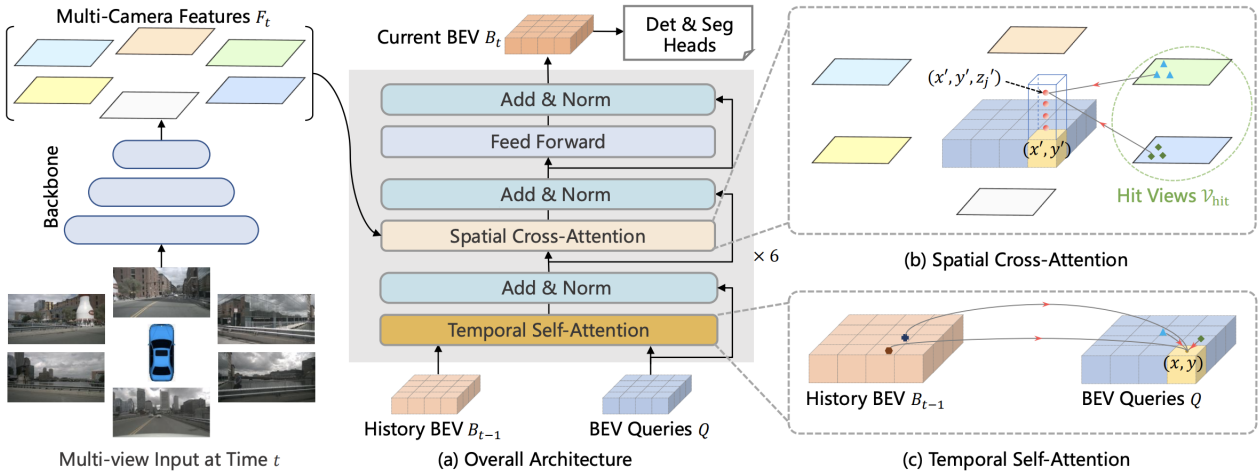


Figure 7: **Overall architecture of BEVFormer** [7]. This model utilize the encoder of Transformer [8] with modification of temporal self-attention and spatial cross-attention.

A.2 SECOND

As depicted in Fig. 8, SECOND (**S**parse **E**mbedding **CON**volutional **D**etection, [66]) consists of two components, one is voxel feature extractor and another is sparse convolutional layers [64]. Firstly, the model convert raw point cloud into voxel as voxel features including coordinates, and then extractor voxel information from voxel feature. Following is a sparse convolutional module, which will efficiently conduct convolution computation, faster the forward speed.

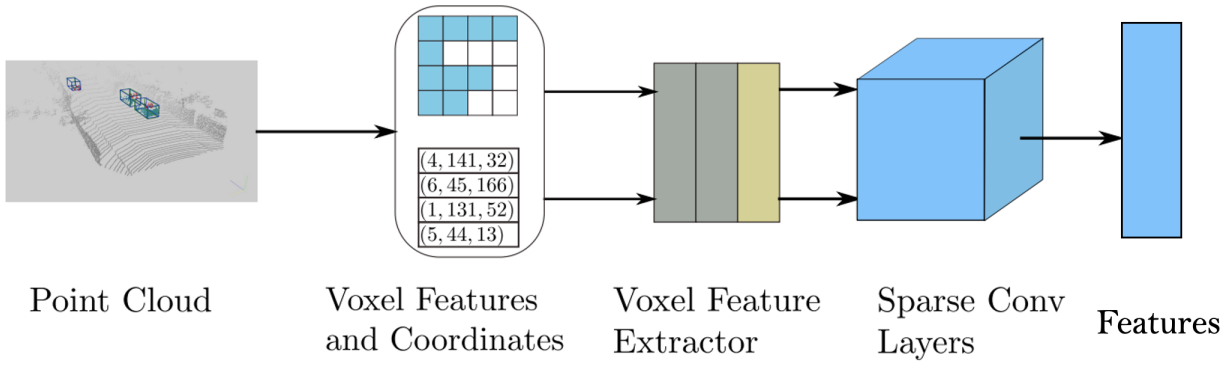


Figure 8: Architecture of SECOND [66]. The highlight of SECOND is sparse convolutional layers, which can significantly improve the computation speed of convolution kernel.

A.3 Pointpillar

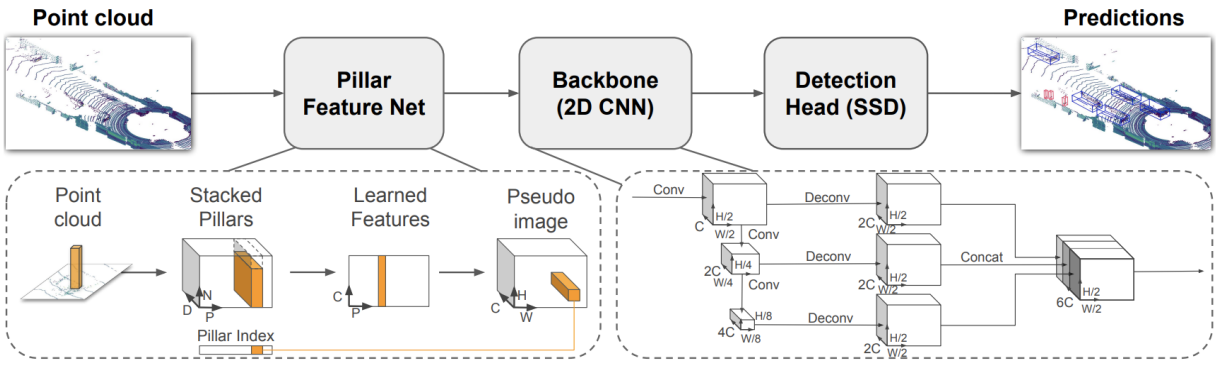


Figure 9: Network structure of Pointpillars [70].

The overall architecture of Pointpillars [70] is shown in Fig. 9. Main idea of this detector is to form a pseudo image so as to feed them into a 2D CNN-style detector. In pillar feature net, raw point cloud is transferred to stacked pillars with a certain amount of points in each pillar. Then the model will extract feature based on stacked pillars, generating a pseudo image. The reason why Pointpillar [70] generate pseudo images is that they can utilize the sophisticated 2D convolutional network and well-designed tricks for training those detectors. In our experiments, we only use the feature generate by Pointpillars, which is the input of SSD detector [20] as depicted in Fig. 9.