BEVENet: Efficient 3D Detection Under Bird's-Eye-View with CNNs

Yuxin Li[1] Qiang Han [2] Mengying Yu[1] Yuxin Jiang[1] Hongxin Wei [1] Bo An[1]
Yiheng Li[2] Zihang Huang [2] Nini Liu [2] Hsuanhan Chen [2] Xiaojun Wu[2]

***Abstract*— 3D object detection in Bird's-Eye-View (BEV) space has become a trending strategy in autonomous driving. Though this paradigm has shown advantages on precision and velocity estimation over perspective view methods, it is nearly impossible to deploy on real-world autonomous driving vehicles. The main reason is the reliance of vision-transformer (ViT) based architecture, which brings in quadratic complexity against input resolution. An efficient design is needed to bypass the drawbacks of ViT models while preserving the powerfulness of BEV-paradigm methods. In this work, we present an efficient BEV-based 3D detection framework, BEVENet. With the convolutional-only architectural design, we demonstrate that CNN architecture is a strong alternative to transformer-based algorithms. BEVENet is 10x times faster than concurrent state-of-the-art methods on NuScenes challenge. With a comparable mAP = 0.416 and NDS = 0.532 on NuScenes val dataset, our inference speed is 25.7 frames per second. To the best of our knowledge, this is the first work considering applying BEV-paradigm methods under limited hardware resources.**

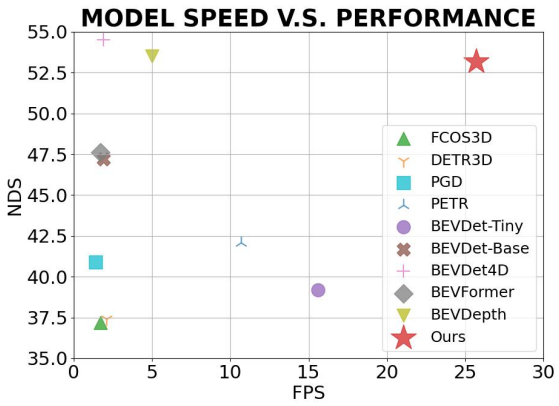


Fig. 1: The comparison of inference speed over different state-of-the-art methods

I. Introduction

3D object detection under Bird's-Eye-View (BEV) space has become a increasingly popular task in the autonomous driving community. As a substitution of the LiDAR-based methods [20, 11, 26], generating pseudo LiDAR points with surrounding cameras is shown to be a promising direction as a cost-effective solution for the autonomous driving subject. In this direction, various methods [13, 9, 8, 12, 17] have been proposed to integrate perception tasks with BEV space representation.

However, the previous methods are generally computationally expensive and rely on large-scale datasets. These conditions, although accessible under the laboratory settings, are challenging to achieve for the in-vehicle environment. More specifically, the vision transformer (ViT) [5] module is one of the major component consuming GPU memory and matrix operations. In the BEV-based methods, vision transformer (ViT) architecture is generally employed due to its capability of capturing semantic information globally. However, this module requires training on large-scale datasets with much more training time than CNN to enable the model to infer the positional relationship between pixels. With a much higher cost for training, ViT only achieves trivial improvements on various vision benchmarks compare to CNN-based models. Another drawback of ViT models is the quadratic complexity over the input dimension, namely the resolution of the input image. Although powerful, it is challenging to deploy on the embedded devices due to the limited computing resources. Moreover, since most objects in autonomous driving scenes are small so that the detection is susceptible to the input resolution. With this constraint, small object detection has been a long-lasting problem of ViT-based models.

Inspired by the above analysis, we propose to overcome the limitation by exploring alternatives, such as pure CNN modules. In this work, we focus on designing an efficient 3D detection framework with the BEV paradigm under limited hardware conditions. We iteratively analysed six classical components in the 3D detection pipeline, namely the backbone, depth estimation, feature projection, temporal fusion, BEV feature extraction and detection head. Model complexity is taken into consideration as it is a critical metric in the real-world deployment of neural network models. Specifically, we propose BEVENet, the abbreviation of BEV-Efficient-Neural-Network, as a computing power friendly model for deployment of the BEV-paradigm method on real-world vehicles. We employ a convolutional-only design to accelerate the inference speed of the model and reduce the memory consumption for large resolution input.

As shown by Figure 1, we demonstrate that pure CNN implementation of BEV architecture is a strong alternative than transformer-based models, with a comparable performance at mAP = 0.416 and NDS = 0.532, our inference speed is 25.7 frames per second, which is 10x times faster and almost 10x times smaller in GFlops than other concurrent state-of-the-art methods on NuScenes challenge.

[1] Nanyang Technological University, Singapore{yuxin004,yume0004,c200203, hongxin001}@e.ntu.edu.sg, boan@ntu.edu.sg
[2] Desay SV Automotive, Singapore {qiang.han, yiheng.li, zihang.huangsgp, nini.liu, hsuanhan.chen, xiaojun.wu}@desaysv.com

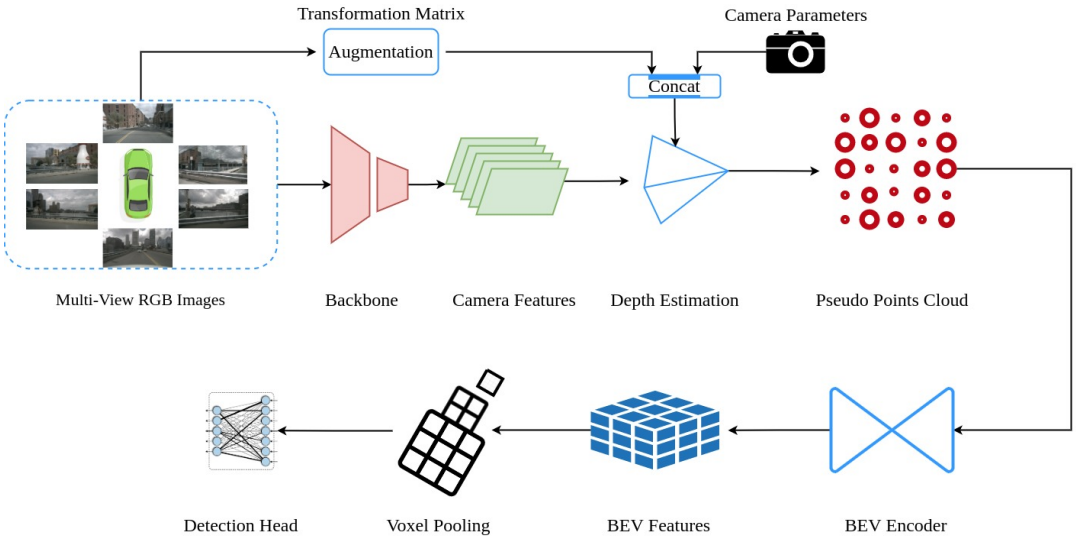


Fig. 2: Pipeline of BEVENet

II. Related work

A. Backbone Models

Backbone models are the foundation of vision perception tasks. Since the induction of AlexNet [10], deep learning has sparked the shining light along the path of fast advancement of various vision tasks. VGGNet [18] and ResNet [6] marked the first two milestone of backbone model in vision. VGGNet first proved the viability of improving vision task performance by increasing the depth of neural networks. ResNet further pushed the boundary of depth of backbone model to 152 layers by the invention of skip-connection. DCN [3] proved that fixed filter size is not necessary in learning vision features. On the side of efficient inference, EfficientNet [19] first brought forward the importance of deployment friendly design while recently RepVGG [4] set a new trending paradigm in inference oriented models. Finally ElanNet [21], an important realization of structural re-parameterization, has proved the exceptional advantage of CNN models in deployment scenarios.

Apart from computer vision, NLP problems have also benefited from the progress of backbone models. Transformers and its variations have dominated the a notable number of NLP tasks. Inspired by the ubiquity of transformers in NLP, Dosovitskiy et al proposed Vision Transformer (ViT) [5] as a universal backbone models for computer vision tasks. Following the insights of dilated convolutional layer, Swin Transformer [16] is another remarkable attempt to apply ViT on vision tasks.

B. 3D Detection Methods

3D object detection, one of the fundamental tasks in perception modules of autonomous driving systems, has witnessed significant progress since the induction of the BEV paradigm. Benefiting from the accurate semantic information provided by surrounding cameras, temporal input from previous frames and supervision over depth information from LiDAR input, camera-only 3D detection methods [12, 13, 9, 8] are approaching their counterpart in LiDAR modality.

DETR3D [24] is the first model that employed such a multi-view image input paradigm to boost the performance of 3D detection. By bringing in temporal information, BEVFormer [13] achieved 10 points higher in mAP than DETR3D. BEVDet4D [9], extending Lift-Splat-Shoot(LSS) [17] with an explicit BEV-Encoder, achieved similar results without the additional input from CAN bus information. BEVDepth [12] added depth supervision over LiDAR input, further proving the effectiveness of perception under BEV space. Lift-Splat-Shoot, a classical yet effective method in projecting image from 2D-view to 3D-view, set the first step in leveraging surrounding cameras input to boost the performance of 3D perception. Each image is "lifted" individually into a frustum of features for each camera, then "splat" all frustums into a rasterized bird's-eye view grid, and finally "shoot" different trajectories onto the cost map. With the spurt of BEV-based methods, we foresee a soon coming date that pure-vision based methods will catch the performance of LiDAR based methods.

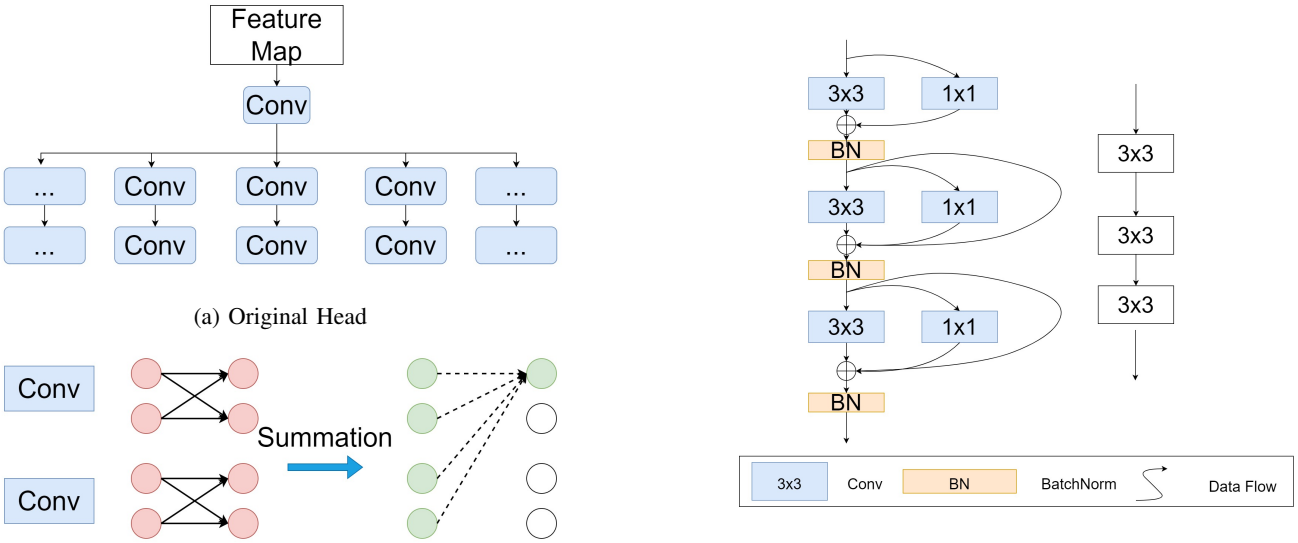
III. Methodology

A. Design Philosophy

We aim to design a efficient model targeting at deployment on limited hardware conditions while preserving the sharpness of BEV-based methods. We adopt a reduction-based methodology, module-by-module we reduce the complexity of the model. We first dissemble SOTA methods on NuScenes challenge leaderboard, then we iteratively combine alternatives of each module and prioritise speed as the benchmark to make the design choice.

B. Network Structure

As shown in Figure 2, the pipeline consists of six modules: a backbone model is shared among multi-view input images; a view projection module to lift flat image input to



(a) Original Head

(b) Merging Multi-Branch Parallel Conv Layers

(c) Merging Conv-BN Layers

Fig. 3: Illustration on Detection Head Simplification by Re-Parameterization

BEV space; a depth estimation module to predict the depth information without LiDAR input; a history caching module to inject temporal information; a BEV feature extraction module to learn the semantic feature under BEV space and finally a detection head.

C. View Projection

Following Lift-Splat-Shoot [17] and BEVDet [9], we set the feature projection module to predict the discrete probability of depth of each pixel. As shown in Figure 4, ground truth of depth is calculated by geometry similarity w.r.t. camera parameters. Let (u, v) be the coordinates of pixel p, the vector value of p in the feature map can be written as $f(u, v)$. The feature vector at position (u, v, d) can be expressed as $alpha \times f(u, v)$, where $alpha$ is the probability of depth d at position $f(u, v)$. Projecting along the light ray, coordinates in the image domain can be converted to spatial domain with camera intrinsic parameters. The final BEV feature is calculated by voxelization at flat surface along x and y directions.

D. Depth Prediction

To address the low accuracy issue of depth estimation from view projection module, another specialised depth estimation module is implemented to enhance the prediction of depth of points. Depth of each point in frustum cloud is superimposed with depth prediction module and fused by weighted averaging. Weights is determined empirically during experiments.

The depth prediction module takes LiDAR points and multi-view images as input. As shown in Figure 5, camera images are first augmented to enhance prediction robustness, and depth information from LiDAR points is treated as ground truth. Image feature and camera parameters, together with the image augmentation transformation matrix, are first concatenated and then fed into an encoding layer. Both camera intrinsic and extrinsic parameters are used to enrich the information prior for the depth estimation module.

E. BEV Encoder

The BEV-Encoder module serves as the intermediate layer to connect pseudo LiDAR cloud and final detection head. Two residual blocks are adopted here to transform the sparse LiDAR points into a dense matrix of feature points. Each grid in the BEV space is generated by voxelization with a pre-defined resolution.

F. Detection Head

The detection head is built on the BEV feature. Based on CenterPoint [25], we set the prediction target to be the position, scale, orientation and speed of objects in autonomous driving scenes. We adopted the same setup as CenterPoint in the training stage for a fair comparison with other concurrent algorithms. At the inference stage, following RepVGG [4], we re-parameterised all multi-branch convolutional layers and batch-norm layers into cascaded plain convolutional networks.

As shown by Figure 3a, the detection head consists of a number of parallel convolutional neural networks. This structure can be simplified by merging convolutional layers and batch-norm layers. As shown by Figure 3b, a Resnet-like architecture is equivalent to a plain convolutional neural network without skip-connection or 1x1 convolution. Identity module is able to directly add to the output feature map without any special operation. While batchnorm layer can be combined with convolutional layer by mathematically summing mean value and standard variance of batch input.

TABLE I: Comparison to state-of-the-art on the NuScenes val dataset

Model	Image Size	GFlops	FPS	mAP	NDS	mATE	mASE	mAOE	mAVE	mAAE
FCOS3D [22]	1600x900	2008.2	1.7	29.5	37.2	0.806	0.268	0.511	1.315	**0.170**
DETR3D [24]	1600x900	1016.8	2.1	30.3	37.4	0.860	0.278	0.437	0.967	0.235
PGD [23]	1600x900	2223.0	1.4	33.5	40.9	0.732	0.263	0.423	1.285	0.172
PETR-R50 [15]	1056x384	-	10.7	31.3	38.1	0.768	0.278	0.564	0.923	0.225
PETR-R101	1408x512	-	3.4	35.7	42.1	0.710	0.270	0.490	0.885	0.224
BEVDet-Tiny [9]	704x256	215.3	15.6	31.2	39.2	0.691	0.272	0.523	0.909	0.247
BEVDet-Base	1600x640	2962.6	1.9	39.3	47.2	0.608	0.259	0.366	0.822	0.191
BEVDepthR50 [12]	704x256	-	11.6	35.1	47.5	0.639	0.267	0.479	0.428	0.198
BEVDepthR101	1408x512	-	5.0	41.2	53.5	0.565	0.266	0.358	0.331	0.190
BEVDet4D-Tiny [8]	704x256	222.0	15.5	33.8	47.6	0.672	0.274	0.460	0.337	0.185
BEVDet4D-Base	1600x640	2989.2	1.9	42.1	54.5	0.579	0.258	0.329	0.301	0.191
BEVFormer [13]	1600x900	1303.5	1.7	41.6	47.6	0.673	0.274	0.372	0.394	0.198
BEVENet(Ours)	704x256	**202.42**	**25.7**	41.6	53.2	0.581	0.272	0.405	0.311	0.194

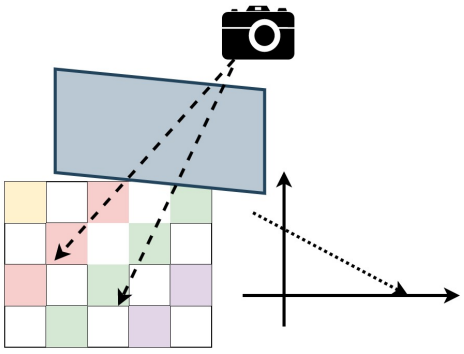


Fig. 4: Illustration of View Projection

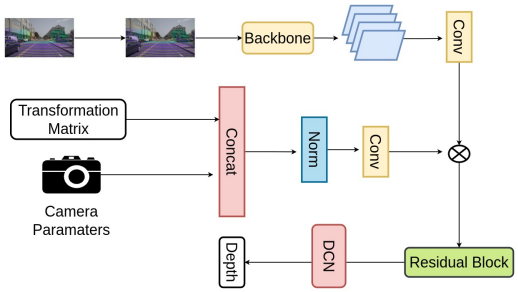


Fig. 5: Illustration of Depth Module. We adopted the same design as BEVDepth [12] in depth estimation module, but adding transformation matrix and extrinsic parameters together with intrinsic parameters as input to depth network. MLP layer is also replaced by a convolutional network.

Formally, let $x_1 = w \times x_0 + b$ and $x_2 = \frac{x_1 - mean}{std}$, then

$$x_2 = \frac{w \times x_0 + b - mean}{std} \tag{1}$$

$$= (\frac{w}{std}) \times x_0 + (\frac{b - mean}{std}) \tag{2}$$

$$= w' \times x_0 + b' \tag{3}$$

Where w' is the new weight combining convolutional layer and batchnorm layer, and b' is the new bias.

As shown by Figure 3c, the multi-branch convolutional module can be simplified by merging weights and bias of neighboring cells. Assume that performance degradation at a single cell prior the merge operation is bounded by post-quantization error E, then the maximum number of cells mergeable n is determined by the pre-quantization or full-precision quantization error e divide by post-quantization error $n = E/e$. The number of neighboring cells to merge is empirically set to two in our experiments.

IV. EXPERIMENTS

A. Experiment Settings

Dataset. We study BEVENet using the public benchmark dataset NuScenes [1]. NuScenes dataset contains 1000 driving scenes, each scene consists of a video clip with 20 second length and each video clip is annotated at the keyframe level sampled at 2HZ. 6 cameras are used to capture images from surrounding views. LiDAR points are captured from 5 Radars and 1 LiDAR. There are in total 10 classes annotated for the 3D detection task: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, barrier and traffic cone. Finally, we set the region-of-interest(ROI) as 51.2 meters in the ground plane with a resolution at 0.8 meters.

Evaluation Metrics We report the performance of experiments according to 2 types of metrics: 1) Official metrics defined in NuScenes challenge including mean Average Precision (mAP), Average Translation Error (ATE), Average Scale Error (ASE), Average Orientation Error (AOE), Average Velocity Error (AVE), Average Attribute Error (AAE) and NuScenes Detection Scores (NDS). 2) Efficiency oriented metrics: Frame Per Second (FPS) and GFlops. FPS is measured with NVidia A100 graphic card, a total of 6019 samples are inferred with standard Pytorch evaluation mode, with preprocessing time excluded. GFlops are measured using the MMDetection3D [2] evaluation toolkit.

Training Parameters Models are trained with AdamW optimizer with learning rate at 2e-4 and weight-decay configured as 1e-2. Batch size are set to be 4 on each of 8 A100 GPU cards. We train for 24 epochs for each round of experiments and save the best model for evaluation.

Data Processing We follow similar setting as BEVDet [9], including notation for processing procedures. Specifically,

TABLE II: Ablation Study

Backbone	S-Head	C-NMS	Depth	2D Loss	Speed(ms)	NDS	mAP	mATE	mASE	mAOE	mAVE	mAAE
Swin-Tiny [16]					56	0.4762	0.3379	0.6714	0.2742	0.4602	0.3371	0.1847
ResNet50 [6]					50	0.4409	0.3329	0.6831	0.2761	0.5444	0.5257	0.2259
Elan [21]					47	0.4903	0.367	0.6965	0.2761	0.4709	0.3246	0.1841
Elan	Y				38	0.4945	0.371	0.6565	0.2745	0.4707	0.3235	0.1846
Elan	Y	Y			**32**	0.4874	0.3663	0.6562	0.2743	0.4682	0.352	0.1840
Elan	Y		Y		43	0.5231	0.4064	0.6072	0.2719	0.4121	0.3257	0.1844
Elan	Y	Y	Y		37	0.5154	0.3923	0.6097	0.2727	0.4133	0.3277	**0.1840**
Elan	Y		Y	Y	43	**0.5318**	**0.4161**	**0.5816**	**0.2719**	**0.4045**	**0.3111**	0.1939

*S-Head stands for whether or not to use simplified head with re-parameterization, C-NMS stands for whether or not to use Circular NMS or Rotate NMS, Depth stands for whether or not to insert depth estimation module, 2D Loss stands for whether or not to add additional 2D supervision on backbone module.

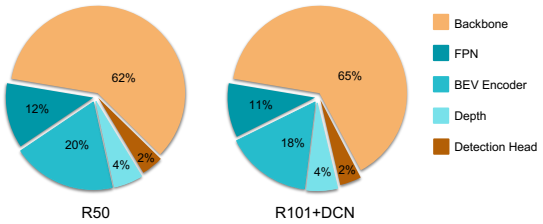


Fig. 6: Percentage of GFlops of Each Module

$W_{in} \times H_{in}$ denotes the input image dimension. The original input size in NuScenes dataset is 1600×900, the processing pipeline includes random flipping, scaling, cropping and rotating. Different from [9], we adopted copy-paste mechanism to balance the skewed distribution of objects. Augmentation operations are traced and converted to a mathematical transformation matrix, which are latter concatenated with camera parameters. Camera parameters are flattened to conform with the fully connected layer dimension. Together with copy-paste mechanism, we adopted Class-Balanced-Grouping-and-Sampling (CBGS) [27] during training, the same method adopted by CenterPoint [25]. During testing phase, scaling operation is applied to ensure the image resolution conforms with the model input. The input image is first scaled by a ratio $s = W_{in}/1600 + 0.4$ and then cropped to size $W_{in} \times H_{in}$ with the region as $(x_1, x_2, y_1, y_2) = (0.5 \times (s \times 1600 - W_{in}), x_1 + W_{in}, s \times 900 - H_{in}, y_1 + H_{in})$.

B. Performance Benchmark

We select 8 SOTA methods on NuScenes leaderboard as our baselines, BEVFormer [13], BEVDet [9], BEVDepth4D [8], BEVDepth [12], PETR [15], PGD [23], FCOS3D [22] and DETR3D [24]. As shown in Table I, we report our best result as mAP = 41.61, NDS = 53.18 on the NuScenes val dataset.

The highest mAP among all methods is BEVDet4D-Base, scores at 42.1. BEVENet is only 1.5 points lower than that. While the highest NDS is also from BEVDet4d-Base, scores at 54.5, which is 1.3 points higher than BEVENet. The main reason for the degeneration of NDS is mAOE of BEVENet is 7 points higher than that of the BEVDet4D-Base. However, with almost same performance, BEVENet achieves 10x times faster than BEVDet4D-Base with almost 15 times less computing power needed. Letting input resolution as 704×256, which is the same size as BEVDet4D-Tiny, BEVENet runs at 25.7 frames per second, with GFlops equals 202.42, while BEVDet4D-Base runs at 1.9 frames per second with GFlops equals 2989.2. Resolution is a critical component in reducing the GFlops of model. BEVDet4D-Base itself is almost 15 times higher GFlops than BEVDet-Tiny, though many other components have also changed, the most prominent factor that incurred this dramatic increase in GFlops is that the input resolution roughly doubled. We reduced the GFlops by first reducing the input resolution and second by replacing transformer based design with convolutional layers.

BEVFormer [13] and BEVDepth [12] are another 2 leading methods on NuScenes 3D detection challenge leaderboard. Compared to BEVENet, BEVFormer achieves the same score in mAP while BEVDepthR101 achieves 0.4 points lower. BEVFormer achieves 5.6 points lower score in mAP while BEVDepthR101 achieves 0.3 points higher. However, both of these 2 methods run much slower than BEVENet. This further proves the effectiveness of our design.

C. Ablation Study

From comprehensive experiments, we finalised the design choices for our efficient model BEVENet. However, it is still unclear which part matters more. For this section we'll report the results and analysis of experiments, to decouple the correlation of each module.

What affects more in GFLops? As the goal of this work is to design an efficient 3D detection framework that can smoothly run under limited hardware resource conditions. It is necessary to analyse which part of the model occupies more computing power. From Figure 6, we can see that backbone module takes over 60% of the GFlops during inference. Naturally, the next question would be how to reduce the GFlops while maintaining good performance. Table I shows that input resolution is the dominant factor of model computing complexity. Since BEV-based methods take six surrounding camera images as input, an intuitive and straightforward strategy to bring down model complexity is to decrease input resolution. However, it can be observed from Table I that directly reshaping the input image will cause a significant drop in performance.

TABLE III: Comparison of Masking Views

Mask	NDS	mAP
∗N/A	35.4	25.22
Front	24.46	19.23
Front Right	23.60	24.42
Front Left	23.65	24.33
Back Right	34.36	23.60
Back Left	34.40	23.94
Back	22.01	17.14
Front Left & Right	23.30	18.92
Back Left & Right	31.72	19.13
Back Three Cams	17.84	10.89
Side Four Cams	16.89	10.98

∗N/A: Baseline model without masking any view.

Which view is more important? Intuitively among 6 surrounding-view cameras mounted on autonomous vehicles, the importance of each view against the ultimate performance of model might be different. BEVFormer [13] and PETR [15] have done robustness analysis on camera intrinsic parameters and randomly dropped images from different views. These works neglect reasoning from the efficient design perspective. From the section above, we know that the majority of GFlops consumption happens at the backbone stage. To this point, it could be an effective method to remove some of the views in surrounding camera input to reduce the GFlops consumption.

As shown in Table III, we find that front and back view camera input are more important than side views. Performance over masking different side views will differ no more than 1 point in mAP. Notably, we find that the NDS is more sensitive against view masking. With front left or front right masked, NDS scores at 23.6 ± 0.05. While masking back left or right view camera input will result in 10 points higher in NDS. We hypothesize that velocity is the major cause. As the NDS metric depends on not only mAP but also the velocity. Front view cameras play a more vital role in providing temporal information. Further experiments on masking both front left and front right view cameras, masking both back left and back right view cameras and masking all three cameras in the back draw the same conclusion. With all results above, we conclude that back left and back right cameras can be masked as a moderate solution to reduce model complexity.

Why didn't FPN work? Feature pyramid network (FPN) [14] has been proven by many works to be an effective module in boosting performance on small and tiny object detection, by incorporating lower level feature maps, region proposal module can generate a finer proposal for objects. However, to our surprise, FPN didn't work well for BEVENet. By increasing number of feature maps from 1 to 4, only a marginal 0.3 points increase in mAP is observed.

We hypothesize that even though small objects comprise a large portion of the NuScenes dataset, the dominating error contribution is not from the scale. Table II further proves our conjecture. Translation error and orientation error both have higher values than scale error. This explains why FPN alone does not improve the overall performance as expected.

What's the impact of backbones? To analyse the difference over backbone models, we picked 3 models to compare, SwinTransformer [16], ResNet [6] and ELanNet [21]. For a fair comparison of results, models SwinTiny, ResNet50 and ELanNet are selected for their similar number of parameters.

As shown in Table II, on the backbone side, ElanNet shows 3.3 points higher in mAP and 2 and 5 points higher than Swin-Tiny and ResNet50. It is a non-trivial problem to coincide a specific task and a performant backbone model. As an inference efficiency driven model, ElanNet has shown a prominent advantage over Swin-Transformer and ResNet50.

What's the benefit of simplified head? On the effect of simplified head, 9ms improvement in speed is observed with replacing the re-parameterized detection head over non-simplified head. Though theoretical operations are mathematically equivalent, a slight increase in mAP and NDS is seen in the comparative experiment. We hypothesize that this is a consequence of scaling operation during inference stage.

Do we need a finely orchestrated NMS? Non-maximum suppression (NMS) [7] is the top concern among all five comparative experiment components. Significant improvements in speed, 6ms faster, can be seen during experiments when replacing Scale-NMS with simple Circular NMS. Apart from the speed metric, no other prominent difference can be seen in the results. This brings us a new question to us, do we need a finely orchestrated NMS? We conclude that even though NMS from different norm may result in possible improvement in precision of model, NMS is not a vital component in our BEVENet compared to the significant faster speed.

Does extra supervision on 2D detection help? 2D perception has witnessed tremendous advancement in recent years, even though there are many differences in 3D and 2D detection tasks. Knowledge prior in design insights among 2D detection tasks can be transferred to 3D-based problems. In this section, we report the effect of adding 2D supervision on backbone model. With 2D loss added, our BEVENet achieved the highest score at 0.416 in mAP and 0.532 in NDS. All subitem metrics outperform the comparative experiment target without 2D loss supervision. We conclude that supervision on 2D loss is an effective design choice.

V. CONCLUSIONS

In this paper, we propose BEVENet, an efficient design of 3D detection framework under constrained real-world autonomous driving hardware systems. BEVENet is developed with convolutional-only layers of neural networks. To the best of our knowledge, this is the first work considering limited hardware resources on real-world autonomous driving vehicles with the BEV paradigm. For future work, we are interested in two more questions 1) Further study on the importance of different channels of image input from multi-view camera setup; 2) Analyse the importance of ROI in BEV space for attributes prediction and model efficiency improvement.

References

[1] Holger Caesar et al. "NuScenes: A Multimodal Dataset for Autonomous Driving". In: *arXiv preprint arXiv:1903.11027* (2019).

[2] MMDetection3D Contributors. *MMDetection3D: OpenMMLab Next-Generation Platform for General 3D Object Detection*. https://github.com/open-mmlab/mmdetection3d. 2020.

[3] Jifeng Dai et al. "Deformable Convolutional Networks". In: *CoRR* abs/1703.06211 (2017). arXiv: 1703.06211. URL: http://arxiv.org/abs/1703.06211.

[4] Xiaohan Ding et al. "Repvgg: Making Vgg-Style Convnets Great Again". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13733–13742.

[5] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ICLR* (2021).

[6] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015).

[7] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. "Learning Non-Maximum Suppression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4507–4515.

[8] Junjie Huang and Guan Huang. "BEVDet4D: Exploit Temporal Cues in Multi-camera 3D Object Detection". In: *arXiv preprint arXiv:2203.17054* (2022).

[9] Junjie Huang et al. "BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View". In: *arXiv preprint arXiv:2112.11790* (2021).

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

[11] Alex H Lang et al. "Pointpillars: Fast Encoders for Object Detection from Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705.

[12] Yinhao Li et al. "BEVDepth: Acquisition of Reliable Depth for Multi-view 3D Object Detection". In: *arXiv preprint arXiv:2206.10092* (2022).

[13] Zhiqi Li et al. "BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers". In: *arXiv preprint arXiv:2203.17270* (2022).

[14] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2117–2125.

[15] Yingfei Liu et al. "Petr: Position embedding transformation for multi-view 3d object detection". In: *arXiv preprint arXiv:2203.05625* (2022).

[16] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[17] Jonah Philion and Sanja Fidler. "Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D". In: *Proceedings of the European Conference on Computer Vision*. 2020.

[18] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[19] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[20] Sourabh Vora et al. "Pointpainting: Sequential Fusion for 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4604–4612.

[21] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors". In: *arXiv preprint arXiv:2207.02696* (2022).

[22] Tai Wang et al. "FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 2021.

[23] Tai Wang et al. "Probabilistic and Geometric Depth: Detecting Objects in Perspective". In: *Conference on Robot Learning (CoRL) 2021*. 2021.

[24] Yue Wang et al. "DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries". In: *The Conference on Robot Learning (CoRL)*. 2021.

[25] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. "Center-Based 3D Object Detection and Tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11784–11793.

[26] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499.

[27] Benjin Zhu et al. "Class-Balanced Grouping and Sampling for Point Cloud 3D Object Detection". In: *arXiv preprint arXiv:1908.09492* (2019).