Advanced Deep Learning for Computer Vision
Center Based 3D Object Detection and Tracking with BiFPN blocks

Dan Halperin
Technical University of Munich
Munich, Germany
dan.halperin@tum.de

Koray Koca
Technical University of Munich
Munich, Germany
koray.koca@tum.de

Abstract

The 3D object-detection problem has a crucial role in understanding 3D environments. However, in contrast to prior research on the 2D domain, it introduces several obstacles, such as biased and expensive point-cloud based datasets, 3D alignment of bounding-boxes and heavy computational load. Therefore, in this project we have chosen to rely on a baseline of CenterPoint, an anchor-free 3D point cloud object detector, and extend it by plugging in a subnetwork of BiFPN convolutional blocks, that aim to process and fuse features from different scales. While the first has achieved remarkable scores on the Waymo and nuScenes benchmarks, we aim to explore what benefits could the BiFPN blocks achieve in terms of object detection and tracking, based on their promising performance in previous works. In our experiments, we found surprising results, such that our proposed architecture generalized better to a wider variety of objects and achieved better predictions for smaller objects such as pedestrians or cyclists. To that end, our code could be reviewed on [6] and [5].

1. Introduction

3D object detection has been an important research topic in recent years. In particular, understanding outdoor 3D environments is crucial for the performance and functionality of safety-critical systems such as autonomous driving.

To this end, an ego-vehicle is equipped with a variety of sensors. The 2D object recognition relies heavily on the images captured by the cameras. However, by projecting data from the 3D physical world onto the image plane, valuable information, such as depth, is lost. Therefore, the 3D variant aims to process and realize 3D point clouds captured by LiDAR sensors. These sensors determine distances by measuring the time it takes for a laser beam to reflect off an object.

In addition, the density of the point cloud varies significantly as a function of distance from the sensor, i.e., points further away from the sensor are very sparsely distributed. On top of that, unlike 2D images, an unordered set of points from a LiDAR scan is more difficult to process.

In this project, we rely on the CenterPoint model [13], which was ranked first in the Waymo benchmark, as a baseline. Based on a strong recognition model, we aimed to improve both applications of CenterPoint, object recognition and tracking, focusing mainly on the first one.

Therefore, we conducted experiments with the original implementation [14] on scenes from the KITTI dataset [3]. The original work appeared as a strong detector for more common objects, such as cars, but was lacking with performance with respect to smaller objects, such as cyclists or pedestrians. Thus, we experimented with integrating BiFPN blocks [11] into the CenterPoint architecture, which process inputs from different scaled feature maps from across the convolutional network and had showed promising results when adopted from other work [2].

2. Preliminaries

2.1. Heatmaps

This is a technique to emphasize the magnitude of activation of objects in a given data-structure. While it is commonly used in data visualization, in this work those heatmaps are used as a tool for 2D detection, where the most activated points are considered to be the center of objects. Typically, a different heatmap is created for each one of the target classes.

2.2. Pseudo-image

This term refers to a 2D data structure that is derived from a 3D structure, by simply merging the depth of 3D voxels with the channels of the 3D feature map:

$$(N, D, C, H, W) \rightarrow (N, D \times C, H, W)$$

An example is demonstrated in Figure 2 (c). These 2D structures can now be processed with efficient 2D convolutions. In this project, a pseudo-image is created in a Bird-eye-view fashion, in order to process the 3D voxel-grid into heatmaps.

2.3. Skip Connections

First introduced by [4], this technique is used in order to solve several problems of deep neural-networks, such as a

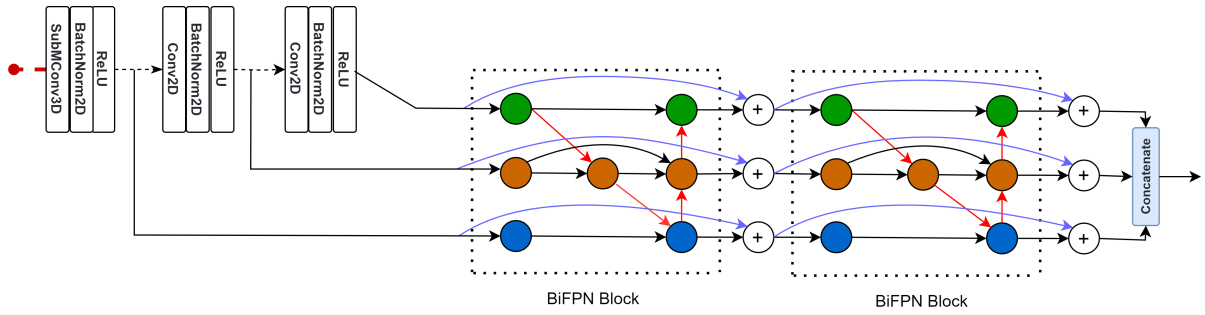

Figure 1. A BiFPN subnetwork, as suggested in [2]. Each block expects 3 inputs from different scales, processes them with convolutional blocks (**colourful nodes**) and performs weighted fusions (**black and red arrows**) in top-down and bottom-up fashion (**red arrows**). Eventually, each block outputs 3 different output streams. The last output streams are concatenated into a single feature map. In addition, the figure demonstrates our integration of skip-connections (**purple arrows**) between the individual blocks, to maintain steady gradient flows, and reusage of former features. Best viewed in color.

vanishing gradient, or allow the reuse of features from previous scales. This is done by the summation of the input and the output of a block of layers. Thus, meaningful gradients can now arrive to the shallow layers of the used network. In this project, these appear to improve our method's performance, allowing us to integrate the BiFPN layers, which are rich with learnable parameters, as shown in Table 2.

2.4. Cyclic Learning Rates

A technique that is used to introduce regularization into the training process to prevent overfitting [9]. The learning rate increases up to an upper bound, towards the mid of the training process, and then drops repeatedly towards zero. This process is done by a trajectory of learning steps that is calculated before the training process begins.

2.5. Focal Loss

As a modified variant of the vanilla cross-entropy loss function [7], it aims to solve the problem of class imbalance by dynamically scaling misclassified examples (a well-known problem for underrepresented classes). Therefore, it is ideally suited for models based on real outdoor 3D datasets. Thus, for an input instance with label t, the loss would be:

$$\mathrm{CE}\left(p_t\right) = -\log\left(p_t\right)$$
$$\mathrm{FL}\left(p_t\right) = -\left(1-p_t\right)^{\gamma}\log\left(p_t\right)$$

Where $\mathrm{CE}\left(p_{\mathrm{t}}\right)$ is the vanilla Cross-Entropy loss function, and $\mathrm{FL}\left(p_{\mathrm{t}}\right)$ is the introduced Focal-Loss. The γ exponent represents the scaling factor.

3. The baseline - CenterPoint

3.1. Object Detection

CenterPoint [13] was proposed in CVPR in 2021 to represent, detect, and track 3D objects within LiDAR point-cloud environments. The prediction of objects in the suggested architecture is done by projection the features from

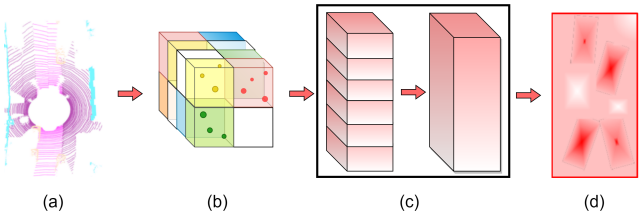

Figure 2. **(a)** Input point cloud. **(b)** Voxelization step with Voxelnet [16]. **(c)** Project to pseudo-image. **(d).** Detection by heatmaps. Best viewed in color.

a 3D voxel grid into a 2D-like pesudo-image (see subsection 2.2), structured with a Bird-Eye-View fashion. Given that, they sought to introduce a model, which is robust to the bounding boxes' alignment with the axes, by detecting objects by their center points, instead of the mentioned boxes. In this way, CenterPoint achieved state-of-the-art results in the nuScenes and Waymo datasets.

3.2. Tracking

The vanilla CenterPoint simplifies downstream tasks such as tracking and uses a greedy closest-point matching method, by utilizing the detected center-points of objects as anchors, instead of the computationally expensive bounding boxes. To this end, it utilizes the detection model. The prediction is made by the relative offset (negative velocity) of objects between consecutive frames, which are then linked up greedily, as shown in Figure 3.

4. Proposed idea - BiFPN

The **BiFPN** (Weighted Bi-directional Feature Pyramid Network) block, was first proposed in [11]. It is a subnetwork of convolutional blocks, that fuses features from different scales of the network, in which it is integrated. In order to do so, it is originally suggested using a pyramid scheme, to extract further features of different scales from a backbone's output. However, [2] proposed a different ap-

Dataset	Enviroment	Scans	Scenes	Split*	Classes	Task	Metric
KITTI [3]	Karlsruhe	14,999	400	0.5/0.25/0.25	11	3D detection	mAP
nuScenes [1]	Boston, Singapore	390,000	1000	0.7/0.15/0.15	23	3D detection and tracking	mAP

Table 1. Overview of different autonomous driving datasets for 3D Detection and tracking challenge. * Split - train/val/train

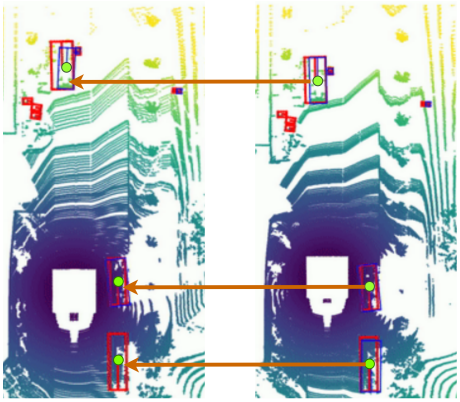

Figure 3. **(a)** Previous frame. **(b)** Current frame. The arrows symbolize the back-projection from the current frame to the previous.

proach, where scales are taken from the stages and blocks of the network itself.

The bi-directionality removes the limits of one-way information flow, with a top-down and bottom-up mechanism, as shown in Figure 1. Since features with different resolutions are fused, they may first need to be resized to the same resolution. The authors of [11] observed that different input features at different resolutions usually contribute to the output feature unequally. To address this issue, they proposed to add another weight for each input, and let the network learn the importance of each input feature.

Moreover, these blocks could be used in a chain, allowing further process to the downstream features, but at a cost of significantly increasing the number of learnable parameters and calculation, while making the vanilla network even deeper.

In addition, each node within a BiFPN block consists of a **convolutional-block**, which in turn is made of a convolution layer, batch-normalization and a ReLU. Those nodes, are then fused by a weighted-fusion mechanism at different steps throughout the block, as proposed in the original paper, to learn the importance of different input features.

Each block has 3 input streams, for 3 different scales, and so is their output. The outputs of the last block in the chain are concatenated together, where they are finally processed together by a convolution block into a single multichannel feature-map.

Our project's guideline is to integrate those blocks into the CenterPoint architecture, and examine their contribution.

5. Experiments and Results

For the detection task, our work is mainly based on the codebase [14]. This repository is based on the one of the official paper [15], by the same authors, originally meant for the nuScenes [1] and Waymo [10] datasets. Our choice of this baseline was made, due to hardware constraints and the fact that the KITTI dataset is smaller. The used pipeline is as show in Figure 2.

As for the hardware, we allocated a virtual machine with Intel N1-highmem-4 (4vCPU, 26 GB of RAM) and an Nvidia Tesla T4 GPU (16 GB of RAM) on the Google Cloud Platform. However, we have encountered several memory issues, which limited us in our ability to perform more meaningful experiments, i.e. deeper BiFPN subnetworks with more consecutive blocks. Also, these hardware bottlenecks resulted with very long training sessions, constraining us to pick only very few configurations to experiment on.

5.1. Experiments

The CenterPoint architecture consists of different backbone convolutional networks (section 3). Our first idea was to plug the BiFPN subnetwork at the end of the 3D-backbone stage, to process the extracted 3D-features directly. In order to mitigate the computational load of those parameters rich blocks, we sought to extract 3 different scales, convert each one into a pesudo-image (subsection 2.2), concatenate and process them through a network of consecutive blocks using GPU-efficient 2D convolutions. Therefore, we expected to obtain a more meaningful and richer single pseudo-image. However, this approach didn't produce an improvement with respect to both the training loss and the validation loss, thus this idea was dropped.

Therefore, our second approach was to implement such a subnetwork at the end of the 2D-backbone stage, that is responsible for the processing of the pseudo-image, as a final step before the creation of the heatmaps, which are the baseline for the detection task. Since this approach had also shown no improvement, we decided to expand the BiFPN subnetwork, by introducing skip-connections (subsection 2.3) between the blocks. Those connections were made by summing each of the 3 outputs of each block, with their corresponding input branch.

With the latter experiment, some improvement could be observed, in terms of training and validation losses alike. Therefore, we started searching for the best hyperparameters. For example, we set the cyclic learning rate (subsec-

model	Recall 0.3	Recall 0.5	Recall 0.7	AP: Cars	AP: Pedestrians	AP: Cyclists	#Parameters (M)
Originial	95.03%	88.44%	65.66%	**96.82**%	70.42%	**92.99**%	**1.880**
BiFPN: [128]	93.67%	87.74%	62.73%	96.27%	52.85%	75.03%	2.913
BiFPN: [128] + SC	**95.44**%	**89.23**%	66.10%	95.31%	**73.02**%	91.60%	2.963
BiFPN: [128, 64] + SC	95.33%	89.16%	**66.68**%	96.45%	**73.02**%	90.24%	3.346

Table 2. Final experiments' results on the KITTI-3D dataset, using 100% of the training data for 120 epochs. SC - Skip connections. Where the BiFPN-integrated model lag in confidence, they compensate with a better generalization to the variety of classes.

tion 2.4), in range of [3e-4, 3e-3]. This approach seemed to perform better in comparison to a fixed learning rate, also sampled from that range. As for the BiFPN blocks, we experimented with convolutions with 64, 96, 128 and 256 channels, separately and in combination. Specifically, we observed two best performing integrations, i.e. a single BiFPN block with 128 channels and a chain of two BiFPN blocks, with 128 and 64 channels, in that order. Furthermore, both architectures utilize skip connections.

It is crucial to mention, that hardware bottlenecks restricted us to experiment with only a few models, and also by their sizes, as the system's memory ran out, more often than not. In addition, for the tracking task we relied on the nuScenes datasets, which is bigger than the KITTI-3D. Due to the limited hardware, we could train our model on 12 epochs using just 5% of training data only. Thus, we could not observe any improvements, since inserting many more parameters into the model, introducing regularization, which in turn requires a longer training session.

5.2. Results

Our detection models were tested by using the "Mayavi" demo platform for point clouds [8], and relying on the open-pcdet library [12], which allows the usage of the official metrics of different benchmarks. Our collected results concern the AP (Average Precision) and recall scores, over 3 classes: Cars, Pedestrians and Cyclists, with a detection confidence threshold of 30%. For the detection task on the KITTI dataset, we have made a few observations: The lower the loss (training and validation), the better the scores for the car class, but worse for smaller objects classes. That could be explained by a few reasons. First, the dataset is flawed, i.e. we observed that the collected point cloud scenes hold more real-world objects than the annotated ones with ground-truth bounding boxes. This pushes the model to overfit over specific geometric shapes. This problem might have been possible to overcome by using the nuScenes dataset, which is much bigger in all aspects, i.e. having much larger distribution of geometrical shapes, for better generalization.

In addition, the used focal loss function (subsection 2.5), which aims to mitigate the imbalanced distribution of different object types, struggles to fairly balance the loss, especially for the baseline model. That could be observed by the fact that as the cars average-scores increase, the ones for pedestrians decrease (Table 2)

Considering the above, we found that the integration with the BiFPN layers outperformed the original implementation of the CenterPoint architecture for the KITTI dataset. While both the original and the modified model suffer from the aforementioned problem, it appears that the BiFPN-based model usually managed to achieve either a higher confidence value for detected objects (mainly cars), but even if it was evaluated with worse confidence, it kept a wider distribution of the objects, i.e. rare objects are detected with relative high confidence, as seen in Figure 4.

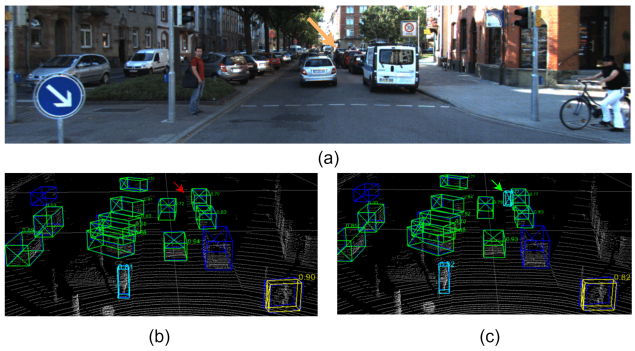

Figure 4. **(a)** Input point cloud from KITTI dataset. Image is given to give a better idea about the scene. **(b)** Detection results of the original network. The pedestrian could not be detected by the original network. **(c)** Detection results of the network with BiFPN. The same pedestrian could be detected by our network. Best viewed in color

6. Conclusions and Future work

We proposed a modified CenterPoint network for simultaneous 3D detection and tracking for objects within LiDAR point-clouds. The suggested BiFPN blocks process and fuse features to improve detection results and generalize over the different classes better than before. However, one should also consider the question - at what cost? The BiFPN blocks introduce hundreds of thousands of learnable parameters, increasing the learning time drastically. We expect that with better hardware and optimization, we could have achieved the improvement in performance, reported in [2].

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020. 3

[2] Zhuangzhuang Ding, Yihan Hu, Runzhou Ge, Li Huang, Sijia Chen, Yu Wang, and Jie Liao. 1st place solution for waymo open dataset challenge – 3d detection and domain adaptation, 2020. 1, 2, 4

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 1, 3

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1

[5] Koray Koca and Dan Halperin. Center-based 3d object detection and tracking with bifpn, 2022. https://github.com/koraykoca/CenterPoint. 1

[6] Koray Koca and Dan Halperin. Center-based 3d object detection and tracking with bifpn - kitti based, 2022. https://github.com/koraykoca/CenterPoint-KITTI. 1

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018. 2

[8] Prabhu Ramachandran. Mayavi:3d scientific data visualization and plotting in python. https://docs.enthought.com/mayavi/mayavi/index.html, 2021. 4

[9] Leslie N. Smith. Cyclical learning rates for training neural networks, 2017. 2

[10] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 3

[11] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020. 1, 2, 3

[12] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 4

[13] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2021. 1, 2

[14] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2021. https://github.com/tianweiy/CenterPoint-KITTI. 1, 3

[15] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking, 2021. https://github.com/tianweiy/CenterPoint. 3

[16] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection, 2017. 2