Learnable Fusion Mechanisms for Object Detection in Autonomous Vehicles

Learnable Fusion Mechanisms for Object Detection in Autonomous Vehicles

Yahya Massoud, Robert Laganiere

***Abstract*— In this work, we propose a novel deep learning based sensor fusion framework, that uses both camera and LiDAR sensors in a multi-modal and multi-view setting. In order to leverage both data streams, we incorporate two new sophisticated fusion mechanisms: element-wise multiplication and multi-modal factorized bilinear pooling. When compared to previously used fusion operators such as element-wise addition and concatenation of feature maps, our proposed fusion methods significantly increase the bird's eye view moderate average precision score by +4.97% and +8.35% for both methods, respectively, when evaluated on KITTI dataset for object detection. Furthermore, we provide a detailed study of important design choices that contribute to the performance of deep learning based sensor fusion frameworks such as data augmentation, multi-task learning, and the design of the convolutional architecture. Finally, we provide qualitative results that showcase both success and failure cases for our proposed framework. We also discuss directions for mitigating failure cases.**

***Index Terms*— Deep Learning, Sensor Fusion, Artificial Intelligence, Autonomous Driving, Object Detection.**

I. Introduction

Modern vehicles comprise a large number of sensors and data streams that can be leveraged to enhance the detection accuracy for autonomous cars' perception system. However, the way this large amount of information coming from different modalities can be combined remains an open problem.

In this paper, we address the problem of multi-modal sensor fusion in the context of designing robust perception algorithms for autonomous driving. Among the different sensors that can be jointly used in an autonomous vehicle, camera and LiDAR is probably the most commonly studied combo. The fact that most available datasets includes these two sensors makes camera-LiDAR fusion system a good candidate in the implementation our fusion network.

Current sensor fusion methods which are depending on both camera and LiDAR offer modest improvement in terms of detection accuracy compared to LiDAR-only methods and this is partly due to using basic fusion mechanisms. We introduce an architecture that takes advantage of a learnable fusion mechanism that has the potential to better exploit the complementarity of the features extracted from both modalities.

II. Related Work

A. LiDAR-only

1) Point-cloud Processing Methods: PointNet [1] was one of the first methods to directly process raw, unstructured, 3D point clouds, for the tasks of classification and segmentation. While PointNet removed the need to transform the point clouds's original representation into a structured format, the method was experimented on with $\sim 1K$ points as inputs, which can be suitable for small sized point clouds, but not for a typical LiDAR scene captured from a car containing $\sim 100K$ points. PointNet++ [2] came to build on top of the aforementioned idea, enabling the processing of overlapped partitions of the input point cloud, giving the architecture the ability to discover hierarchical structures.

2) Volume-based Methods: VoxelNet [3] was proposed to operate directly on input point clouds by using a voxelization technique. The main innovation was passing the non-empty voxels to a sequence of Voxel Feature Encoding (VFE) layers. While the paper showed impressive detection accuracy, it lagged the aspect of inference speed, as it relied on a computationally expensive 3D convolutional architecture. On the other hand, PointPillars [4] introduced a novel point cloud encoding system that simplifies the use of an efficient 2D convolutional architecture for detection and classification. The idea is to separate the space into pillars instead of voxels, then encode these pillars to produce a top-view pseudo image, which is then fed it into a detection architecture. The end-to-end architecture produced state-of-the-art (SOTA) results while being efficient in terms of computational complexity, and faster in terms of frames-per-second (FPS).

3) Projection-based Methods: PIXOR [5] is a single-stage architecture, designed to produce bird's eye view (BEV) bounding box detections by processing projected top-view point-clouds. The idea is to construct a 3D occupancy grid that discretized the physical space ($L \times W \times H$). The spatial dimensions encode the width and the length of the physical space, while the height is encoded in the channel dimension. PIXOR showed great improvement in the real-time performance. PIXOR++ [6] improved the network architecture, as well as the output encoding of the bounding boxes, to enhance both the detection accuracy and the speed of the framework.

R. Laganiere is professor at the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, Canada, K1N 6N5 (e-mail: laganier@uottawa.ca).

Y. Massoud was with the University of Ottawa. He is now AI Software Developer at Sensor Cortek inc., Canada.

The authors wish to thank Tianran Liu for his contribution in the experimental validation of the MFB fusion mechanism.

FVNet [7] relied on the cylindrical frontal view projection of the point clouds to perform the detection task. It is worth noting that projection-based methods tend to suffer from information loss due to discretization.

B. Camera-only

Images can be processed in multiple ways, and many features could be extracted from them. Mono3D [8] leveraged many of these features, such as: context, semantic and instance segmentation, shape and location priors, etc. Mono3D is a two-stage monocular 3D object detection system, which generates 3D region-proposals from the input image, then passing these proposals into a Fast R-CNN-like [9] architecture for producing the final bounding boxes. PseudoLiDAR [10] adopted the idea of using a depth estimation module that generates a dense depth map from an image to create a *pseudo*-LiDAR point cloud. Combining this method with a region proposal network, and a 3D instance segmentation network, PseudoLiDAR is capable of producing 3D bounding boxes from a 2D input image.

C. Camera-LiDAR

Several works have attempted to improve perception by combining camera and LiDAR information. To combine such information, two main design decisions have to be taken: i) at which point in the network should the two sensors be combined and; ii) which operator should be used to fuse the data extracted from both sensor streams.

1) Where to fuse: Early fusion aims to fuse the input modalities before performing any feature extraction step. Many fusion-based methods start by feeding input modalities to feature extractors, such as: [11]–[15]. Intermediate fusion as in [11], [15] is performed on a feature map level, meaning that the modalities have been already processed by many feature extraction layers. Late fusion fuses high-level feature maps right before the final detection layers.

2) How to fuse: The choice of a fusion mechanism is an important factor towards building a robust sensor fusion architecture as it enables the interaction between different modalities and their extracted features. Basic and computationally efficient fusion methods can be used, such as: element-wise summation, mean, max, or the concatenation operation. However, these methods are not robust enough to capture related features between different modalities. [16] introduced bilinear-pooling as a robust fusion mechanism. Despite the expressiveness of bilinear pooling, it is impractical to use it from a computational perspective. [16]–[18] present more efficient ways to implement factorized versions of the robust bilinear-pooling fusion mechanism in deep learning architectures.

3) Sensor Fusion Methods: MV3D [11] presents a sensor fusion framework that leverages both camera and LiDAR sensors. Moreover, the architecture uses cylindrical frontal view projection of the LiDAR sensor to further enhance the performance of its frontal view branch. Each of the modalities is fed into a feature extraction network, then a 3D region-proposal generation step is applied. After that, region proposals are fed into a deep-fusion mechanism to enable interaction between all three modalities. Finally, the output of the deep-fusion mechanism is fed to classification and regression branches. It is worth noting that the deep-fusion mechanism comprises of a branching version of the element-wise mean fusion. Frustum PointNets [12] uses RGB-D data for the tasks of object detection and localization. The framework applies a robust 2D detector to the RGB image to extract accurate 2D region-proposals, then aims to extract and segments 3D frustums from each region proposal by using the depth information. PointFusion [13] is a two-stage framework that uses inputs from both camera and LiDAR. PointFusion applies a SOTA 2D detector on the image, then fuses the extracted feature maps with the corresponding 3D points extracted from the raw LIDAR point cloud to mitigate any loss of information, not having to encode the whole raw point cloud. AVOD [14] incorporates a two-stage architecture to operate on both camera and LiDAR sensors, while the intermediate fusion mechanism is an element-wise mean operation, followed by a series of fully-connected layers for each modality. ContFuse [15] is an end-to-end framework that uses the idea of continuous convolution to project frontal view feature maps extracted from a frontal view stream into BEV, then fuses these projected feature maps with original BEV feature maps extracted from a BEV feature extractor. The fusion mechanism is a simple element-wise summation.

We propose here a camera-LiDAR sensor fusion approach based on a mid-level fusion architecture. We demonstrate that using a multi-modal bilinear fusion operator constitutes an effective way to fuse feature maps of different modalities. Basically, this fusion mechanism allows the network to learn how the feature maps coming from the two sensors can be optimally combined.

III. Methodology

A. Input and Output Representation

1) Input Representation: The inputs of our fusion architecture come from a camera sensor and a lidar sensor.

Camera Sensor. RGB images provide our sensor fusion framework with color and texture information to enhance the capability of the feature extraction module. Each RGB image has a size of $L \times W \times C$, where L and W represent the image's spatial dimensions, and C representing the number of channels. One of the disadvantages of using the camera only is its vulnerability to adverse light and weather conditions, and their lack of depth information. That is the reason why we decided to augment our framework's input with the LiDAR sensor.

LiDAR Sensor. LiDAR point-clouds are unstructured and sparse. In this work, we use a projected version of the raw 3D LiDAR points to enhance the detection accuracy for our sensor fusion framework. We first adopt the top-view projection of the point-cloud, i.e. bird's eye view (BEV). We do that by discretizing the physical dimensions constructing an occupancy cuboid. We also discretize the height and encode it into the channels dimension to reduce information loss. In addition, we project the 3D LiDAR points into the frontal view. We fuse the frontal view projection with the RGB

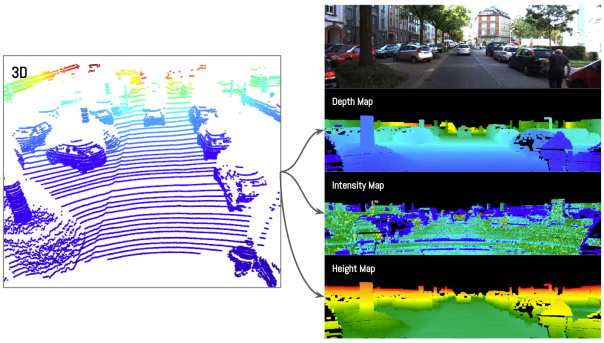

Fig. 1. LiDAR point clouds' 3D to frontal view projection of the following features: depth, height, and intensity.

image early in the framework. The features we extract from the frontal view projection are intensity, depths, and height maps. Such LiDAR features are very useful in our case, as they complement the camera's lacks of such information. The projected LiDAR features are visualized in Figure 1.

2) Output Representation: **Bird's Eye View.** We use 3D LiADR points to generate the framework's output encoding. This is done by projecting the ground-truth 3D bounding boxes to the top-view to construct a boolean heatmap. Moreover, we ignore boundary pixels of the top-view groudtruth bounding boxes by generating a zoomed version of these boxes, and we call this process *subsampling*. The usage of *subsampling* is inspired by Pixor architecture [5]. The overall downsampling factor of the top-view boolean heatmap is 2 It is worth noting that we use top-view output encoding for two main reasons: (1) to simplify the detection task as all the objects are perceived from top-view, hence, there will be no occluded objects, and (2) all the target class objects (e.g. cars) will have similar sizes from the top-view, which simplifies the bounding box estimation task. As for the regression targets, we also encode them into a top-view representation, to match the spatial dimensions of the top-view objectness map. We only consider a bounding box if its corresponding confidence score in the objectness map exceeds a specific threshold (e.g. 0.5). We encode each bounding box information into center coordinates $\{x_c, y_c, z_c\}$, object's size $\{w, l, h\}$, which are then normalized to be $\log(w), \log(l), \log(h)$, and finally, the heading angle factorized into $\cos(\theta), \sin(\theta)$. The end-to-end architecture aims to learn all of these 8 regression values. **Frontal View.** We only generate objectness maps for the frontal view. We use such 2D objectness maps to enhance the quality of the feature maps generated by frontal view stream. The downsampling factor of the 2D output encoding is 4.

B. Fusion Architecture

1) Factorized Bilinear Pooling: The multi-modal factorized bilinear pooling (MFB) learnable fusion mechanism was proposed in [16] as a fusion technique used for temporal activity detection in videos. In our work, we adapt this robust fusion mechanism for the object detection task in the field of autonomous driving perception systems. Our objective was to design a fully convolutional architecture that leverages sophisticated fusion mechanisms. To enable maximum interaction

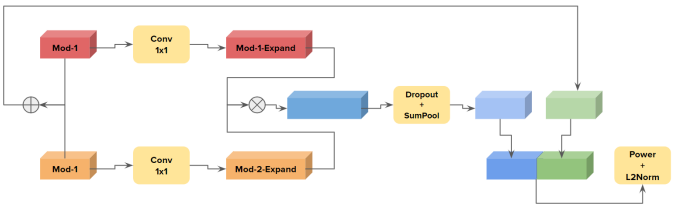

Fig. 2. Architecture of Multi-modal Factorized Bilinear Pooling.

between both input modalities we perform the following steps: (1) we project both modalities into a higher dimensional space using 1x1 convolutional layers (as opposed to fully connected layers that were used in the original implementation of [16]), (2) both projected feature maps are combined using element-wise multiplication, followed by two operations, drop-out and sum pooling, (3) the original input modalities are concatenated with the resulting feature map, (4) power and l_2 normalization are applied. Figure 2 illustrate the fusion mechanism in a block diagram.

2) Frontal View Stream: The frontal view stream consists of a series of convolutional blocks acting as a feature extractor whose inputs are the frontal view representations: the monocular RGB images and the frontal view projection of the 3D LiDAR points. Both modalities should have the same spatial dimensions. First, we perform an **early** fusion operation for both modalities using the multi-modal factorized bilinear pooling mechanism as explained in Figure 2. This fusion operation combines the richness of RGB images with the expressiveness of LiDAR features to produce a more robust representation. The feature extraction network has the following structure: three ResNet-blocks [20], a total downsampling factor of 8, with the number of layers in each block as $\{2, 4, 4\}$, respectively. The number of filters is multiplied by 2 each time we perform a downsampling operation. The number of filters per layer is $\{32, 64, 128\}$, respectively. We use strided convolutions as the main downsapling operation, and ReLU activation is used for non-linearity. We employ an FPN-like [21] architecture to combine the output feature maps of each convolutional block. This produces a final feature map with a downsampling factor of 4, so we resize the feature maps of both the first and third blocks using a bilinear resizing operation. After each resizing operation we apply 1×1 convolution to mitigate the checkerboard effects. Finally, we concatenate all three feature maps in the channels' dimension, and we feed it to an intermediate fusion module. The final concatenated feature map is also used as a training target for the 2D frontal view objectness task.

3) Bird's Eye View Stream: The bird's eye view stream is employed in parallel with the frontal view stream. It processes an occupancy cuboid through three ResNet-based [20] convolutional blocks, each containing $\{4, 6, 6\}$ layers, respectively. The number of filters per block is $\{32, 64, 128\}$, respectively. ReLU is used as the main non-linearity operation. Downsampling is applied through strided convolution, and the overall downsapling factor of the BEV stream is 8. As in the frontal-view stream, we combine the multi-scale BEV feature maps to finally produce a feature map with a total

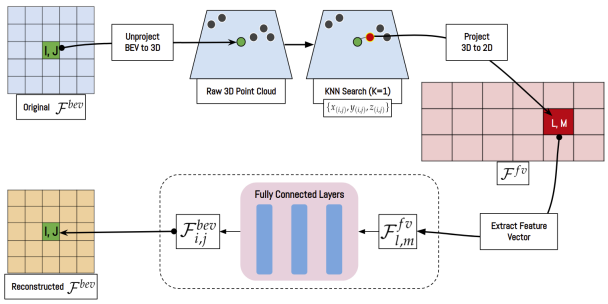

Fig. 3. Overview of the Continuous Convolution process.

downsampling factor of 2.

4) Learnable Fusion Mechanism: In our work, we propose to fuse both frontal and top view modalities together. Such a transformation process is not trivial, hence, we use a Continuous Convolutions module, initially proposed in [15]. The idea is to let the end-to-end framework learn a mapping between frontal view and top view feature maps. The process can be summarized as follows:

1) Step 1: We iterate through the BEV feature map's ($\mathcal{F}^{bev}$) spatial dimensions to unproject each pixel from BEV to 3D using the provided camera-LiDAR calibration parameters.
2) Step 2: For each position in the BEV map ($\mathcal{F}^{bev}_{i,j}$), we search for its closest neighbouring unprojected 3D point.
3) Step 3: For each unprojected 3D point, we project it back to the image plane (frontal-view $\mathcal{F}^{fv}$) to obtain indices $\{l, m\}$. These indices are then used to query the frontal-view feature map $\mathcal{F}^{fv}$.
4) Step 4: The outcome of the query (a feature vector $\mathcal{F}^{fv}_{l,m}$) will then be fed into a series of fully connected layers, aiming to reconstruct the corresponding BEV feature vector $\mathcal{F}^{bev}_{i,j}$r.

The continuous convolution process is applied to the frontal stream multi-scale feature maps. The resulting feature map of this process is resized to three different scales to match the sizes of the output feature maps from the BEV stream. Then, **intermediate** fusion is applied to the transformed FV feature maps and the corresponding BEV feature maps using the Multi-modal Factorized Bilinear Pooling mechanism. Finally, the three fused feature maps are scaled to the same size, concatenated in the channels' dimension, and fed into the final detection heads. The aforementioned process is illustrated in Figure 3.

5) Detection Heads: **Classification Head.** We use a sigmoid-activated 1×1 convolution operation to generate objectness maps with the help of the original boolean heatmaps constructed from the groudtruth data. **Regression Head.** We use a linearly-activated convolutional layer with eight filters to estimate the bounding box information. The complete architecture of our network is illustrated in Figure 4.

C. Training

1) Objective Functions: We adopt a multi-task loss for in our sensor fusion architecture. The multi-task loss consists of three components: frontal-view classification loss, top-view classification loss, and top-view regression loss, as shown in Equation 1. We use the focal loss [22] for binary classification (Equation 2) and smooth-L1 loss for regression (Equation 3).

$$\mathcal{L}_{multitask} = \mathbf{FL}_{Front} + \mathbf{FL}_{BEV} + \mathbf{L1\text{-}smooth}_{BEV} \quad (1)$$

$$\begin{aligned} p &= \begin{cases} \hat{y}, & \text{if } y = 1 \\ 1 - \hat{y}, & \text{otherwise} \end{cases} \\ \mathbf{FL}(p) &= -\tfrac{1}{N}(\alpha(1-p)^{\gamma}\log(p)) \\ &\alpha\text{: weight factor} \\ &\gamma\text{: modulating factor} \\ &N\text{: number of positive pixels} \end{aligned} \quad (2)$$

$$\mathbf{L1\text{-}smooth} = \begin{cases} 0.5(y_i - \hat{y}_1)^2/\beta, & \text{if } |y_i - \hat{y}_1| < \beta \\ |y_i - \hat{y}_1| - 0.5 * \beta, & \text{otherwise} \end{cases} \quad (3)$$

2) Data Augmentation: Despite being a benchmark for many object detection tasks, the KITTI [19] is, according to nowadays standards, of relatively small size which could lead to overfitting. To address this issue, we employed multiple multi-modal data augmentation techniques applied on both LiDAR and camera inputs simultaneously. The data augmentation strategies include: horizontal flipping, per-box dropout of 3D points, global dropout of 3D points, scaling, and translation for both vertical and horizontal axes.

D. Inference

Frontal- and top-view projections are generated from LiDAR with the help of the camera calibration parameters. We crop the 3D point cloud based on the camera's field-of-view (FoV). Input modalities are then fed into the fusion model to obtain objectness and regression maps. These maps are then decoded to produce estimated 3D bounding boxes. A candidate box is decoded from the regression map only if its corresponding confidence score is above a 50% threshold. Finally, non-maximum supperssion (NMS) is applied to remove duplicate detections using intersectoin over union (IoU). Our end-to-end sensor fusion framework simultaneously produces accurate bounding box detections in 2D, 3D, and BEV.

IV. Experiments

A. Evaluation of Sensor Fusion on KITTI

1) Experimental Setting: We perform our experiments on KITTI [19] dataset by evaluating our model's performance for 2D, 3D, and BEV object detection tasks simultaneously. We rely on KITTI's validation set to showcase our experimental results, by following the split suggested in [23]. We used the validation set to enable flexibility in the analysis of our results, which is advantageous compared to using the test set only through KITTI's benchmark server. We train our framework on the "Car" class, and evaluate it using the average precision (AP) metric, at intersection-over-union (IoU) threshold of 0.5.

The input shapes of our deep learning architecture are based on the original size of the RGB images, $1242 \times 375 \times 3$, then we apply min-max normalization on the RGB values. Each of the

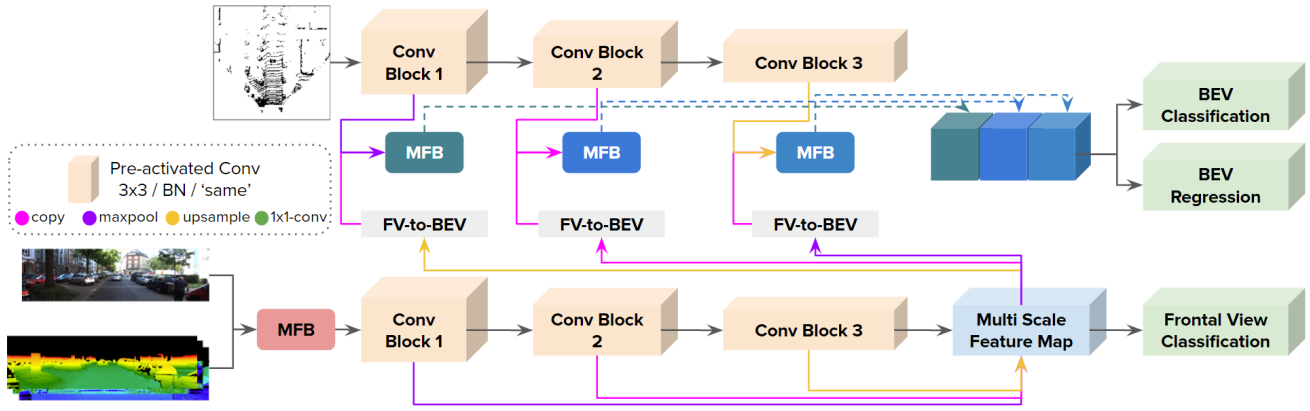

Fig. 4. The proposed camera-lidar fusion architecture.

LiDAR projected features have the same spatial dimensions as the RGB images, however, each contain only a single channel. All three LiDAR projected feature maps are concatenated in the channel dimension, forming another $1242 \times 375 \times 3$ input that can be easily concatenated with the RGB image. The BEV occupancy map is constructed by defining a 3D physical space, $L \times W \times H$, as $[0, 70] \times [-40, 40] \times [-1, 3]$, respectively, and by defining discretization factors $[0.15625, 0.15625, 0.125]$ for each of the axes, respectively. We use these values to produce a BEV image with spatial dimensions 448×512, along with 32 channels to encode the height information. Based on the architectural design mentioned in Section III, the framework produces frontal view objectness maps of size $310 \times 94 \times 1$, and BEV objectness and regression maps of size $224 \times 256 \times 1$ and $224 \times 256 \times 8$, respectively. The deep learning architecture only consists of pre-activated residual convolutional blocks [24], unless stated otherwise.

We train using RMSProp optimizer with a batch size of 2 and an exponentially decaying learning rate schedule, starting with 0.0005 for the first 30 epochs and a decay factor of 0.8 every epoch, then a learning rate of 0.0003 for the final 30 epochs. The parameters for the focal loss are $\alpha = 0.75$ and $\beta = 1.0$, and the smoothing factor of the L1 loss is $\sigma = 3.0$. We use online data augmentation strategy (applied during training), while employing 2 NVIDIA GeForce GTX 1080Ti in parallel. We use TensorFlow framework for training and TensorBoard for visualization.

2) Evaluation Metric: We use KITTI's official evaluation kit [27], which computes the average precision (AP%) metric. to indicate the detection accuracy of an object detector. The evaluation kit uses 40 recall points [26] to compute AP and produces final detection accuracy for *easy*, *moderate*, and *hard* cases. We report AP with $IoU = 0.5$, unless stated otherwise.

3) Early Fusion Evaluation Results: A critical element in our proposed framework is the early fusion of LiDAR projected features (intensity, height, and depth maps) and RGB images in the frontal view stream. Simply put, without using these LiDAR frontal view features, we have color and texture information, while lacking depth, intensity, and height information from the scene. This type of information is important to accurately estimate 3D oriented bounding boxes. In Table I, we show significant improvements in AP% in all three difficulty levels and in all three detection tasks (2D, 3D, and BEV), where BEV indicates LiDAR BEV projection, RGB indicates input image to the frontal view, and Li-FV indicates LiDAR projected frontal view features. A consistent improvement in the hard category can be concluded from fusing Li-FV with RGB, resulting in (+5.01%) increase in BEV detection, (+8.09%) in 3D detection, and (+2.67%) in 2D detection.

Input Modalities	Easy	Moderate	Hard
	BEV Detection		
BEV + RGB	92.45%	86.48%	81.77%
BEV + RGB **+ Li-FV**	**95.16%** (+2.71)	**89.12%** (+2.64)	**86.78%** (+5.01)
	3D Detection		
BEV + RGB	88.01%	77.04%	72.33%
BEV + RGB **+ Li-FV**	**91.75%** (+3.74)	**82.94%** (+5.90)	**80.42%** (+8.09)
	2D Detection		
BEV + RGB	82.8%	80.71%	78.5%
BEV + RGB **+ Li-FV**	**87.26%** (+4.46)	**83.13%** (+2.42)	**81.17%** (+2.67)

TABLE I
SHOWING THE IMPACT OF COMBINING RGB IMAGES WITH USING LIDAR'S FRONTAL VIEW PROJECTIONS.

4) Learnable Fusion Mechanism Evaluation Results: This subsection highlights our most important contribution; the impact of using sophisticated fusion mechanisms to enhance the overall detection accuracy of a sensor fusion framework. We focus on the intermediate fusion between frontal view and bird's eye view streams. Since the final detections are estimated from the outcome of this fusion stage, the fusion operators can either be a bottleneck or a booster of the performance. In Table II, we show the significance of the proposed fusion operator. Our baseline for intermediate fusion is the element-wise (EW) addition operation, which was widely adopted in previous methods like [15].

We also highlight the replacement of EW addition with EW multiplication, which resulted in a significant increase in the detection accuracy in all tasks and all difficulty levels. This modification resulted in (+6.44%), (+4.97%), and (+2.84%) increase in the BEV task compared to EW addition, in easy, moderate, and hard cases, respectively. Similar trends can be found for both 2D and 3D detection tasks as well. We argue that the reason for such performance increase is the nature of EW multiplication operator, which enables solidifying the high-valued features through the subsequent layers, and dimin-

ishing low-valued features. We can think of it as a ReLU [25] layer that yields high performance for mutli-modal inputs.

The second and most important experiment concerns the use of a learnable fusion mechanism, the multi-modal factorized bilinear pooling (MFB-Pooling), for the intermediate fusion task. The MFB-Pooling module outperformed the EW multiplication method, resulting in (+2.92%) for easy cases, (+3.38%) for moderate cases, and an important (+6.16%) for hard cases. Learnable fusion mechanisms introduce a robust, yet efficient, way to enable maximum adaptation and interaction between multi-modal inputs, to produce the best fusion function that has the lowest generalization error. Although this method adds up more learnable parameters compared to fixed fusion operators, we showed that it can be implemented in a fully-convolutional manner that will result in a unified and robust sensor fusion framework.

Intermediate Fusion Type	Easy	Moderate	Hard
	BEV Detection		
EW-Addition	85.8%	80.77%	77.78%
EW-Multiplication	92.24%	85.74%	80.62%
MFB-Pooling	**95.16%**	**89.12%**	**86.78%**
	3D Detection		
EW-Addition	68.61%	59.75%	57.29%
EW-Multiplication	87.14%	76.64%	71.38%
MFB-Pooling	**91.75%**	**82.94%**	**80.42%**
	2D Detection		
EW-Addition	71.12%	69.14%	68.93%
EW-Multiplication	84.32%	79.71%	77.37%
MFB-Pooling	**87.26%**	**83.13%**	**81.17%**

TABLE II
SHOWING THE IMPORTANCE OF USING INTERMEDIATE LEARNABLE FUSION MODULES TO BOOST DETECTION ACCURACY.

5) Further Validation: In order to further assess the validity of our learnable fusion mechanism, we incorporated the MFB-Pooling module with a previously proposed sensor fusion framework. The plug-and-play nature of this fusion module makes it easy to experiment with other sensor fusion frameworks aiming to further enhance their performance.

We conducted experiments on AVOD [14] which employs an intermediate fusion mechanism into its Region Proposal Network (RPN). AVOD feeds both input modalities (3D point cloud and RGB image) into separate feature extractors, then fuses the resulting feature maps using element-wise mean operation to generate region proposals. We replace the element-wise mean operation used in AVOD with our proposed learnable fusion mechanism. In Table III, we observe that the version of AVOD employing the MFB-Pooling consistently outperforms the original implementation in 3D and BEV detection tasks.

B. Ablation Study

1) Data Augmentation: The use of data augmentation techniques during training process is of utmost importance, especially when using KITTI data set, which is limited in terms of the diversity and the number of available driving scenarios. We developed and integrated data augmentation strategies that are applied on both input modalities. The augmentation operations include translation, up- and down-scaling, random dropout of 3D points, and flipping along the x-axis. Each augmentation operation had a probability of being applied in each training step (e.g. applying scaling 30% of the time). This enabled us to potentially combine multiple augmentation strategies in a single step, which results in an ever-increasing number of unseen examples during training. Observing Table IV, it is clear that data augmentation increases the framework's ability to generalize. Performance boost in hard cases can be listed as a (+10.73%) in BEV detection, (+11.07%) in 3D detection, and (+11.41%) in 2D detection.

Architecture	Easy	Moderate	Hard
	BEV Detection		
AVOD w/o MFB	88.67%	79.33%	78.27%
AVOD w/ MFB	**89.72%** (+1.05)	**80.16%** (+0.83)	**79.11%** (+0.84)
	3D Detection		
AVOD w/o MFB	77.40%	67.33%	65.66%
AVOD w/ MFB	**78.36%** (+0.96)	**68.20%** (+0.87)	**66.44%** (+0.78)

TABLE III
EVALUATING THE IMPACT OF INTEGRATING THE PROPOSED MFB LEARNABLE FUSION WITH AVOD [14] ARCHITECTURE.

	Easy	Moderate	Hard
	BEV Detection		
w/o Data Aug.	87.65%	80.30%	76.05%
w/ Data Aug.	**95.16%** (+7.51)	**89.12%** (+8.82)	**86.78%** (+10.73)
	3D Detection		
w/o Data Aug.	83.48%	73.27%	69.35%
w/ Data Aug.	**91.75%** (+8.27)	**82.94%** (+9.67)	**80.42%** (+11.07)
	2D Detection		
w/o Data Aug.	67.29%	69.88%	69.76%
w/ Data Aug.	**87.26%** (+19.97)	**83.13%** (+13.25)	**81.17%** (+11.41)

TABLE IV
COMPARING THE PROPOSED ARCHITECTURE WITH AND WITHOUT APPLYING DATA AUGMENTATION TECHNIQUES.

2) Multi-task Training: Forcing the deep learning architecture to learn diverse tasks simultaneously also contributes in boosting the detection accuracy. In our framework, we add a new term to the objective function that enables the network to predict 2D heatmaps from the frontal view stream. The extra output branch helps the network learn high quality frontal view feature maps. In table V, we observe that applying this modification increases the performance, resulting in (+5.33%) for BEV detection, (+11.86%) for 3D detection, and (+5.10%) for 2D detection.

Learning Tasks	Easy	Moderate	Hard
	BEV Detection		
w/o FV. Objectness	93.09%	86.26%	81.45%
w/ FV. Objectness	**95.16%** (+2.07)	**89.12%** (+2.86)	**86.78%** (+5.33)
	3D Detection		
w/o FV. Objectness	85.87%	73.35%	68.56%
w/ FV. Objectness	**91.75%** (+5.88)	**82.94%** (+9.59)	**80.42%** (+11.86)
	2D Detection		
w/o FV. Objectness	85.81%	80.33%	76.07%
w/ FV. Objectness	**87.26%** (+1.45)	**83.13%** (+2.8)	**81.17%** (+5.1)

TABLE V
COMPARING THE MODEL GENERALIZATION CAPABILITY WHEN OPTIMIZING FOR MULTIPLE SUPERVISED LEARNING TASKS.

3) Convolutional Block Design: Another important component is the inner design of the convolutional blocks. We experimented three types of convolutions: (1) separable convolutions, (2) standard convolutions, and (3) pre-activated residual convolutions. The sophisticated design of the pre-activated residual convolutions [24] proved its robustness, as it enables a direct linear connection throughout the whole architecture, not just within a local convolutional block. This enables more interaction between different feature maps within the multi-stream architecture. Hence, we conclude the importance of interactability between different stream in sensor fusion frameworks. Table VI further demonstrates pre-activated residual blocks consistently outperform its counterparts.

Conv. Type	Easy	Moderate	Hard
	BEV Detection		
Separable Conv.	77.92%	71.03%	64.7%
Standard Conv.	85.47% (+7.55)	78.96% (+7.93)	74.67% (+9.97)
Pre-activated Residual Conv.	**95.16%** (+9.69)	**89.12%** (+10.16)	**86.78%** (+12.11)
	3D Detection		
Separable Conv.	65.77%	54.47%	49.35%
Standard Conv.	74.25% (+8.48)	60.29% (+5.82)	59.06% (+9.71)
Pre-activated Residual Conv.	**91.75%** (+17.50)	**82.94%** (+22.65)	**80.42%** (+21.36)
	2D Detection		
Separable Conv.	64.38%	65.29%	64.68%
Standard Conv.	63.83% (-0.55)	65.56% (+0.27)	65.24% (+0.56)
Pre-activated Residual Conv.	**87.26%** (+23.43)	**83.13%** (+17.57)	**81.17%** (+15.93)

TABLE VI
CONTRASTING DETECTION ACCURACY WHEN EMPLOYING DIFFERENT TYPES OF CONVOLUTIONAL BLOCKS.

C. Qualitative Results

In this subsection, we systematically analyze and study qualitative results produced by our proposed sensor fusion framework to highlight its success and failure cases, further understand causes, and propose potential solutions. Each visualization will have the following structure: (1) 3D bounding boxes drawn on RGB images in the top left, (2) projected 3D bounding boxes drawn on RGB images in the bottom left, (3) projected BEV bounding boxes drawn on the BEV projection of the raw LiDAR 3D points on the right. The groundtruth boxes are presented in the green color, while the predictions are presented in the red color.

1) Extreme Occlusion: Occlusion is arguably one of the most challenging problems facing perception systems. Even in the case of leveraging multiple sensor modalities, extreme occlusion can still negatively impact the detection accuracy, as shown in Figure 5. To mitigate such an issue, we can employ a tracking module that takes into consideration previous frames, where an occluded object could have been detected.

2) Rare Scenarios: The issue of facing rare cases emerge from the lack of enough driving scenarios in KITTI [19], and an example of that can be found in Figure 6. Such a challenge can be addressed by training the framework on large-scale data sets that contains diverse and sophisticated driving scenarios.

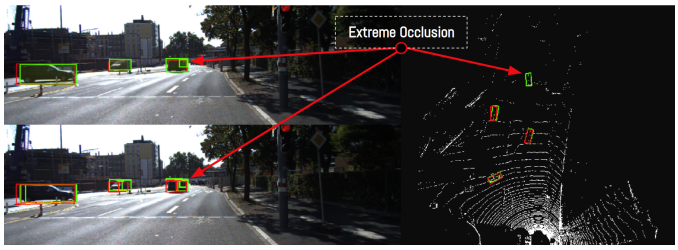

Fig. 5. A scenario from KITTI [19] where a car is highly occluded with respect to both Camera and Lidar. GT: green, PRED: red.

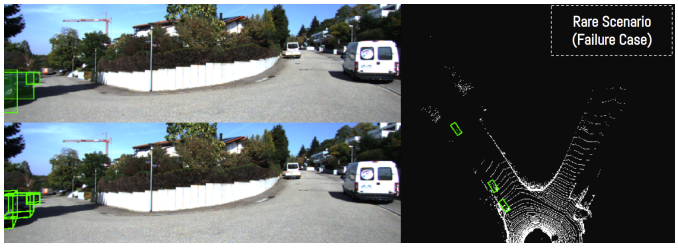

Fig. 6. A rare scenario from KITTI [19] that is tricky to our fusion network and yeilds a failure case. GT: green, PRED: red.

3) Success Cases: Our sensor fusion framework shows robust capability in detecting moving and non-moving cars in different scenarios as in Figures 7 and 8.

Fig. 7. A scenario from KITTI [19] shows our network's high detection accuracy for moving cars. GT: green, PRED: red.

Fig. 8. A scenario from KITTI [19] showing our network's high detection accuracy for non-moving cars. GT: green, PRED: red.

4) Discussion: In Figures 9 and 10, and 11, we see a fundamental issue with KITTI [19] dataset, where many cars where not annotated and are not considered as groundtruth. Here, we have detected 6 unannotated cars in 2 different frames, and since they are counted as *false positives* instead of *true positives*, they contribute to a drop in the average precision metric. Producing a well annotated data set is a challenging and time consuming task, but the quality of the training data

is of an utmost importance for producing robust and accurate deep learning models.

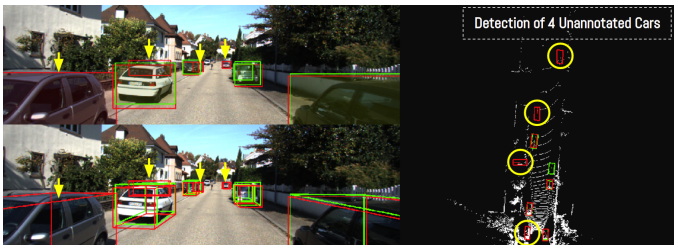

Fig. 9. A scenario from KITTI [19] where four unannotated cars were correctly detected by our fusion network. GT: green, PRED: red.

Fig. 10. A scenario from KITTI [19] where two unannotated cars were correctly detected by our fusion network. GT: green, PRED: red.

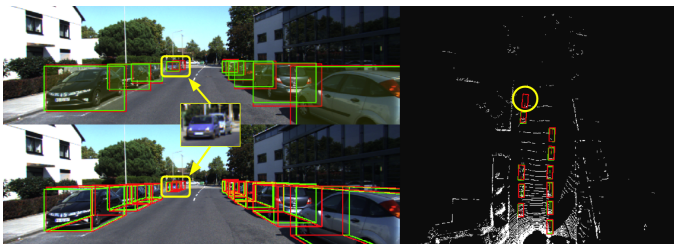

Fig. 11. A scenario from KITTI [19] showing our network's high detection accuracy when capturing unannotated cars. GT: green, PRED: red.

V. Conclusion

In this work, we illustrated the process of designing robust sensor fusion frameworks, as well as highlighting the impact of employing sophisticated mechanisms as early and intermediate fusion techniques. Moreover, we supplied our contributions with ablation study that covers important aspects of fusion architectures. Fusion architecture is a promising direction to produce robust multi-modal perception systems. However, the design and training of such architectures remain challenging. While we adapt a learnable fusion mechanism to the object detection task, we believe that this is just scratching the surface of a family of sophisticated, robust, learnable fusion approaches. We believe that using such learnable fusion mechanisms can lead to more reliable detection networks.

References

[1] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77-85, doi: 10.1109/CVPR.2017.16.

[2] Ruizhongtai Qi, Charles Yi, Li Su, Hao Guibas, Leonidas. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.

[3] Zhou, Yin Tuzel, Oncel. (2018). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. 4490-4499. 10.1109/CVPR.2018.00472.

[4] Lang, Alex Vora, Sourabh Caesar, Holger Zhou, Lubing Yang, Jiong Beijbom, Oscar. (2019). PointPillars: Fast Encoders for Object Detection From Point Clouds. 12689-12697. 10.1109/CVPR.2019.01298.

[5] Yang, Bin Luo, Wenjie. (2018). PIXOR: Real-time 3D Object Detection from Point Clouds. 7652-7660. 10.1109/CVPR.2018.00798.

[6] Yang, Bin Liang, Ming. (2020). HDNET: Exploiting HD Maps for 3D Object Detection.

[7] Zhou, Jie Tan, Xin Shao, Zhiwen Ma, Lizhuang. (2019). FVNet: 3D Front-View Proposal Generation for Real-Time Object Detection from Point Clouds. 1-8. 10.1109/CISP-BMEI48845.2019.8965844.

[8] He, T., Soatto, S. (2019). Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors. ArXiv, abs/1901.03446.

[9] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[10] Wang, Yan Chao, Wei-Lun Garg, Divyansh Hariharan, Bharath Campbell, Mark Weinberger, Kilian. (2019). Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. 8437-8445. 10.1109/CVPR.2019.00864.

[11] Chen, Xiaozhi Ma, Huimin Wan, Ji Li, Bo Xia, Tian. (2017). Multi-view 3D Object Detection Network for Autonomous Driving. 6526-6534. 10.1109/CVPR.2017.691.

[12] C. R. Qi, W. Liu, C. Wu, H. Su and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 918-927, doi: 10.1109/CVPR.2018.00102.

[13] D. Xu, D. Anguelov, A. Jain, 'PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation', 11 2017.

[14] Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L. (2018). Joint 3D Proposal Generation and Object Detection from View Aggregation. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1-8.

[15] H. Hu, J. Beck, M. Lauer, C. Stiller, 'Continuous Fusion of Motion Data Using an Axis-Angle Rotation Representation with Uniform B-Spline', Sensors, . 21, . 15, 2021.

[16] Rahman, M., Laganière, R. (2020). Mid-level Fusion for End-to-End Temporal Activity Detection in Untrimmed Video. BMVC.

[17] J.-H. Kim, 'Hadamard Product for Low-rank Bilinear Pooling', ArXiv, 2017.

[18] Yu, Zhou Yu, Jun Fan, Jianping Tao, Dacheng. (2017). Multi-modal Factorized Bilinear Pooling with Co-attention Learning for Visual Question Answering. 1839-1848. 10.1109/ICCV.2017.202.

[19] A. Geiger, P. Lenz, R. Urtasun, 'Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite', Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[20] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[21] Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J. (2017). Feature Pyramid Networks for Object Detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 936-944.

[22] T. -Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2999-3007, doi: 10.1109/ICCV.2017.324.

[23] Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., Urtasun, R. (2018). 3D Object Proposals Using Stereo Imagery for Accurate Object Class Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40, 1259-1272.

[24] He, Kaiming Zhang, Xiangyu Ren, Shaoqing Sun, Jian. (2016). Identity Mappings in Deep Residual Networks. 9908. 630-645. 10.1007/978-3-319-46493-0_38.

[25] Nair, Vinod Hinton, Geoffrey. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. Proceedings of ICML. 27. 807-814.

[26] A. Simonelli, S. R. Bulò, L. Porzi, M. L. Antequera and P. Kontschieder, "Disentangling Monocular 3D Object Detection: From Single to Multi-Class Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 3, pp. 1219-1231, 1 March 2022, doi: 10.1109/TPAMI.2020.3025077.

[27] Carlos Guindel (2017) eval_kitti. Source Code. https://github.com/cguindel/eval_kitti