A Simple and Efficient Multi-task Network for 3D Object Detection and Road Understanding

Di Feng[1,2], Yiyang Zhou[1], Chenfeng Xu[1], Masayoshi Tomizuka[1], Wei Zhan[1]

Abstract— Detecting dynamic objects and predicting static road information such as drivable areas and ground heights are crucial for safe autonomous driving. Previous works studied each perception task separately, and lacked a collective quantitative analysis. In this work, we show that it is possible to perform all perception tasks via a simple and efficient multi-task network. Our proposed network, LidarMTL, takes raw LiDAR point cloud as inputs, and predicts six perception outputs for 3D object detection and road understanding. The network is based on an encoder-decoder architecture with 3D sparse convolution and deconvolution operations. Extensive experiments verify the proposed method with competitive accuracies compared to state-of-the-art object detectors and other task-specific networks. LidarMTL is also leveraged for online localization. Code and pre-trained model have been made available at **https://github.com/frankfengdi/LidarMTL**.
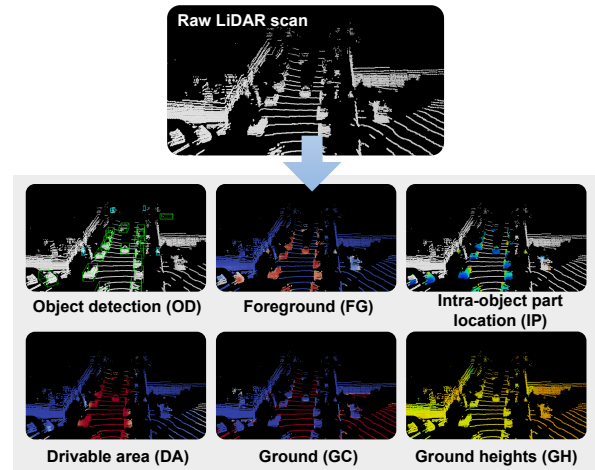
Fig. 1: The proposed multi-task network, LidarMTL, takes raw Lidar point clouds as inputs, and performs six 3D perception tasks in one forward pass in a frame-by-frame manner. The tasks include object detection (OD), foreground point classification (FG), intra-object part location regression (IP), object-free drivable area classification (DA), ground area classification (GC), ground height estimation (GH).

I. INTRODUCTION

Reliable traffic object detection and road understanding near the ego-vehicle are fundamental perception problems in autonomous driving. Movable objects are often perceived at the instance level with class labels and bounding boxes, or at the 2D image pixel or 3D point level depending on sensing modalities. A comprehensive road understanding requires the perception algorithm to identify drivable areas, lane markings, and road shapes, to name a few. Furthermore, all these perception tasks need to run accurately and quickly for online deployment. However, most existing methods, especially those using deep learning approaches, focus on improving each task separately, with task-specific network architectures and evaluation metrics. This task-specific solution is inefficient when dealing with multiple tasks. While high inference speed might maintain via parallel computing, the memory footprints and computation costs scale linearly with the number of networks, which quickly become infeasible with limited hardware resources.

Multi-Task Learning (MTL) provides a strategy to largely reduce memory footprints and computation costs by performing all tasks via a unified model in one forward pass [1]. In deep learning, MTL translates to learn the shared representation of multiple tasks typically via an encoder-decoder network architecture. MTL was applied to 2D object detection and road understanding using RGB camera images [2]–[4] and has been recently introduced to 3D perception using Lidar point clouds [5]–[7].

In this work, we propose a Lidar-based multi-task learning network called LidarMTL to jointly perform six perception tasks for 3D object detection and road understanding, as shown by Fig. 1. Objects are detected with class labels and 3D bounding boxes (task OD). Furthermore, their associated Lidar points are segmented as foreground (task FG), and their relative locations to object centroids are regressed (task IP). Road perception includes point-wise drivable area and ground area classification (tasks DA and GC) as well as ground height estimation (task GH).

The benefits of those perception tasks have been studied in previous works. For example, FG and IP are leveraged to refine bounding boxes in the second-stage of a two-stage object detector [8], as they provide useful point-level semantic and geometrical information of objects. GC and GH are used to remove ground [9] or normalize the heights of Lidar points [6], especially when the ground is not flat. DA provides strong road priors to reduce false-positive predictions in object detection [5] and motion forcasting [10]. Unlike previous works which explore each perception task separately, we show that it is possible to perform all perception tasks efficiently and accurately via a unified network. Besides, previous works such as [6], [8] focus on how to employ one or several perception tasks as auxiliary tasks to

[1] Mechanical Systems Control Lab, University of California, Berkeley, CA, 94720, USA.
[2] Institute of Measurement, Control and Microtechnology, Ulm University, 89081, Ulm, Germany.
Correspondence: di.feng@berkeley.edu

support the target task, without analyzing the performance of those auxiliary tasks. In this work, we consider each perception task of equal importance, and conduct comprehensive experiments to analyze their performance in the single-task and multi-task settings.

In principle, the LidarMTL network works by adding task-specific heads to a 3D UNet architecture and training the full network with a multi-task loss in an end-to-end manner. UNet is a well-performed encoder-decoder network widely applied to 2D image segmentation. Following [8], we extend UNet to efficiently process 3D Lidar points represented as voxels with 3D sparse convolution and deconvolution operations. The resulting network has only 6.5M trainable parameters and runs at an inference speed of 30FPS on a Titan RTX GPU, which is 2× smaller and 6× faster than performing all tasks sequentially using task-specific networks. Extensive experiments on the Argoverse Dataset [10] shows that the LidarMTL network achieves competitive accuracies compared to state-of-the-art object detectors and other task-specific networks. The network is also employed to substantially improve online localization.

II. RELATED WORKS

A. Lidar-based Object Detection

Lidar point cloud is usually represented by 2D projected images [11], [12], raw Lidar point [13], [14], and voxels [15]. Compared to the other methods, voxel representation can not only be processed efficiently using 3D sparse convolution [16], but also preserve approximately similar information to raw point cloud with small voxel size. Therefore, voxel-based backbone networks have been widely applied to learn Lidar features in conjunction with 2D CNN detection head [8], [16]–[18]. A special case is PointPillars [19], which efficiently processes Lidar points by vertical 3D columns called pillars. Our proposed LidarMTL network follows this "voxel-based backbone + 2D CNN detection head" pipeline to perform object detection.

B. Road Understanding

Understanding the 3D road information online is crucial for safe autonomous driving, especially when HD maps are not available. A variety of methods have been proposed for online mapping, such as road area classification [20], [21], lane and boundary detection [22], [23], ground plane estimation [24], road topology recognition [25], [26], and road scene semantic segmentation [27]–[29]. In [7], a multi-task network is designed for multiple object-free road perception tasks, including drivable area classification, road height estimation, and road topology classification.

C. Joint Object Detection and Road Understanding

Existing methods usually follow the hard parameter sharing scheme [1], where networks consist of a shared encoder and several task-specific decoders. MultiNet [2] jointly performs object detection, street recognition, and road area classification. It is built by a large 2D CNN encoder based on the VGG16 or ResNet backbones, followed by task-specific branches with several convolution layers. DLT-Net [3] follows the similar architecture for object detection, road area

classification, and lane detection. Besides, HDNet [5] and MMF [6] propose to use drivable road maps or ground heights as auxiliary inputs for Lidar-based object detectors, in order to improve the detection accuracy by adding road priors. Our proposed LidarMTL also uses the hard parameter sharing: a detection head and a decoder with sparse deconvolutions are added to the encoder for object detection and point-wise predictions, respectively.

III. METHODOLOGY

A. Task Definition

The proposed Lidar-based multi-task learning network, LidarMTL, jointly performs six perception tasks via a single feed-forward pass, namely, 3D object detection (OD), foreground classification (FG), intra-object part location regression (IP), drivable area classification (DA), ground area classification (GC), and ground height regression (GH). The method is developed based on the Argoverse dataset [10], because to our knowledge it is the only public dataset that provides both dynamic object labels and static map information with ground heights and drivable areas.

More specifically for these tasks, Lidar points within the bounding boxes of the dynamic objects are regarded as foreground. Intra-part object locations are defined as the positions of *foreground* points relative to their corresponding object's centroids. Driveable areas are *object-free* regions which could be driven by vehicles. Ground height estimation is performed both for ground areas and non-ground areas (such as foreground and buildings).

B. Overview

We aim to design a simple and efficient multi-task network for joint 3D object detection and road understanding. In this regard, the LidarMTL is based on the voxelized Lidar point cloud representation and the UNet backbone with 3D sparse convolution and deconvolutions (we name the model "UNet3D"). Fig. 2 shows the network architecture. The 3D space is voxelized into regular voxels, with no-empty voxels being encoded with Lidar features. The voxelized Lidar point cloud is processed by the UNet backbone network with the encoder-decoder architecture [30]. The encoder consists of several 3D sparse convolutions, and downsamples the input spatial resolution by 8 times in order to extract high-level Lidar features. The decoder gradually upsamples the Lidar features to the original spatial resolution via 3D sparse deconvolutions. We choose the UNet backbone network and voxelized Lidar representation following the idea from [8]. The network well-preserves the geometric information of Lidar points by setting a proper voxel size, and has been shown in [8] to achieve higher efficiency than the raw point-based methods (such as PointRCNN [13]).

Point-wise predictions are made by adding output layers directly on the decoder network, including tasks FG, IP, DA, GC, and GH. To perform object detection (OD), the 3D Lidar features from the encoder are projected onto the Bird's Eye View (BEV), and then processed by a detection head with several standard 2D convolution layers for classification score prediction and bounding box regression. Note that
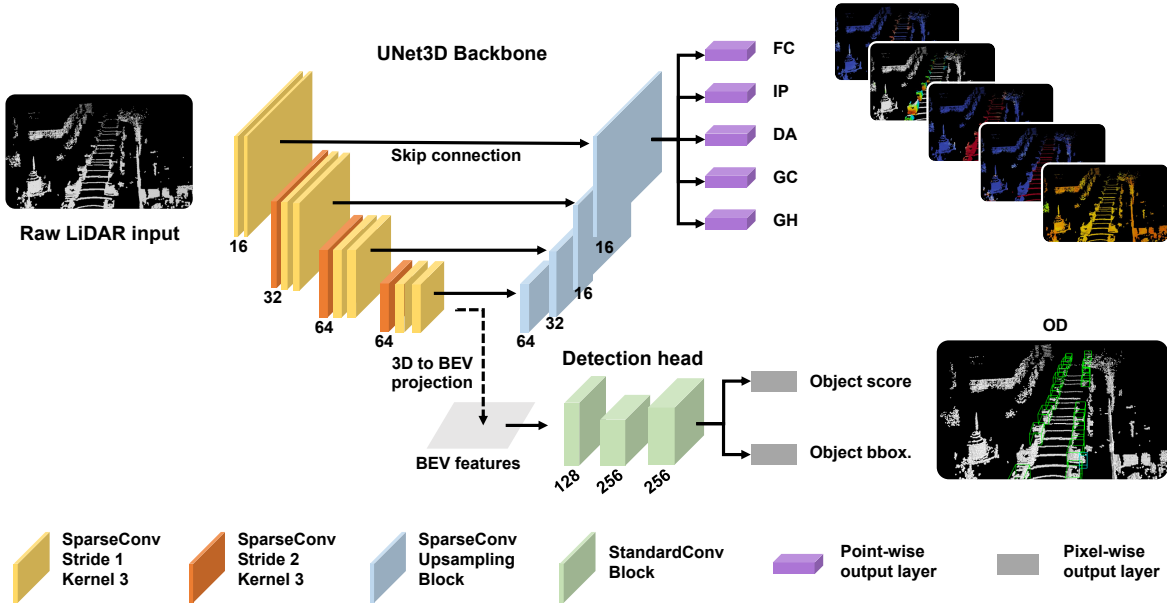
Fig. 2: The LidarMTL network architecture. The network jointly performs object detection (OD) and five point-wise perception tasks, namely, foreground classficiation (FC), Intra-object part location regression (IP), drivable area classification (DA), ground classification (GC), and ground height estimation (GH). The network is based on a UNet backbone with 3D sparse convolution and deconvolution. A small detection head with 2D convolution is added to perform object detection on the Lidar Bird's Eye View (BEV).

employing 2D CNN on Lidar BEV features is a common way to do object detection [16], [17], [19]. Besides, it is found more effective than performing object detection from the decoder network (cf. Sec IV-B.1).

C. Input and Output Representation

Given an Lidar scan, let y be the target output. The input features of each voxel are encoded as the mean values of the Lidar point positions in the Lidar coordinate system. The perception tasks FC, DA, and GC are formulated as the point-wise binary classification problems, with their labels $y_{FG}, y_{DA}, y_{GC} = 1$ indicating positive samples, and 0 negative samples. The tasks IP and GH are formulated as the point-wise regression problems, with $y_{IP} = [x', y', z']$ a continuous vector indicating 3D Lidar point locations relative to their corresponding object centroids, and y_{GH} the ground heights. As for OD, the label y_{OD} consists of object classes y_{cls}, and bounding box regression variables y_{bbox}, i.e. $y_{OD} = [y_{cls}, y_{bbox}]$. Bounding boxes are parameterized by $y_{bbox} = [\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h, \Delta \theta]$, with $\Delta x, \Delta y, \Delta z$ being the residual centroid 3D positions, $\Delta l, \Delta w, \Delta h$ the residual length, width, and height, and $\Delta \theta$ the residual orientation relative to pre-defined anchors. The network makes predictions for $y_{FG}, y_{DA}, y_{GC}, y_{cls}$ via the softmax function, and directly regresses the bounding box parameters. Following [8], y_{IP} are normalized to be within $[0, 1]^3$ and are predicted by the softmax function as well, as this encoding strategy is found more stable than direct regression.

D. UNet3D Backbone

As shown in Fig. 2, the encoder in the backbone processes the voxelized Lidar point cloud by four stages of

3D sparse convolutions with increasing feature dimensions $16, 32, 64, 64$. The network downsamples the spatial resolution by 8 times through three sparse convolution layers [16] with stride 2, each followed by two submanifold sparse convolution layers [31] with stride 1. The decoder consists of four upsampling blocks with decreasing feature dimensions $64, 32, 16, 16$ and strides $2, 2, 2, 1$. In each block, the features from the previous block are combined with the skip-connected features from the encoder via concatenation, and are further processed by a submanifold sparse convolution layer and a sparse inverse convolution layer, in order to upsample the spatial resolution. All convolution and deconvolution layers in the backbone have a kernel size of $3 \times 3 \times 3$. Finally, task-specific $1 \times 1 \times 1$ sparse convolution layers are added to the last upsampling block for point-wise predictions.

E. Detection Head

The detection head projects the 3D Lidar features from the UNet3D encoder to the Bird's Eye View (BEV), and processes the BEV features through three 2D convolution blocks. The first block consists of six standard convolution layers with feature dimension 128 and stride 1. The second block increases feature dimension to 256. It downsamples the spatial resolution by a convolution layer with stride 2, stacked with five 2D convolution layers with stride 1. The last block is an upsampling layer with dimension 256 and stride 2. All convolution layers in the detection head have a kernel size of 3×3. The classification scores and bounding boxes are predicted by the output layers with 1×1 convolution.

Similar to [16], the object detector regresses residual

bounding box parameters relative to the pre-defined 3D anchors with fixed sizes, because objects from the same category are of approximately similar sizes. For each pixel and object category, we place two anchors with rotations of 0 and 90 degrees, with their sizes being the mean values from all ground truths in the Argoverse dataset.

F. Joint Training

The full network is trained end-to-end via a multi-task loss function. Denote L as a loss function. For an input data frame, the multi-task loss function, L_{MTL}, is formulated as a weighted sum of the task-specific losses:

$$L_{\mathrm{MTL}} = \sum_{\substack{i \in \{\mathrm{OD,FG,IP,} \\ \mathrm{DA,FC,GH}\}}} w_i L_i, \tag{1}$$

where w_i and L_i represent the task-specific loss weights and loss functions, respectively. To learn $y_{\mathrm{DA}}, y_{\mathrm{GC}}, y_{\mathrm{IP}}$, we use the standard cross entropy loss. As for y_{FG} and y_{cls}, we use the focal loss [32] due to the large positive-negative sample imbalance problem. Finally, y_{GH} and y_{bbox} are learnt by the standard L_1 loss.

A loss weights w_i controls the influence of a task. It can be pre-defined through grid search, or optimized by task balancing approaches [1]. In this work, we employ the uncertainty weighting strategy proposed by Kendall *et al.* [33]. It uses the task-dependent uncertainty, parameterized by the noise parameter σ, to balance the single-task losses. Such noise parameters are jointly optimized during training, resulting in an adaptive multi-task loss function $L_{\mathrm{MTL}}^{\mathrm{adaptive}}$ written as:

$$L_{\mathrm{MTL}}^{\mathrm{adaptive}} = \sum_{\substack{i \in \{\mathrm{OD,FG,IP,} \\ \mathrm{DA,FC,GH}\}}} \frac{1}{2\sigma_i^2} L_i + \frac{1}{2} \log \sigma_i^2 \ . \tag{2}$$

IV. EXPERIMENTAL RESULTS

The experimental results are structured as follows. In Sec. IV-B, we evaluate the performance of each perception task separately. We compare the proposed multi-task network with single-task networks, and show its benefits in achieving on-par performance with task-specific networks, but with substantially lower memory footprints and higher inference speed. Afterwards, we conduct ablation studies in Sec. IV-C regarding the number of tasks and the loss weights, and test the network's robustness with downsampled Lidar points. Finally, we demonstrate in Sec. IV-D that our proposed multi-task network provides useful semantics which largely improve online localization.

A. Experimental Setup

1) Dataset

All experiments are conducted on the Argoverse 3D Tracking Dataset [10], which was recorded in Miami and Pittsburgh in the USA under various weather conditions and times of a day. The dataset provides 3D bounding boxes and tracks annotations, with RGB images from seven cameras, Lidar point clouds from two 32-beam Velodyne Lidar sensors, as well as HD maps annotating drivable areas, ground heights,

ground areas and center-lines. For the object detection task, we focus on the "VEHICLE" and "PEDESTRAIN" classes. For the point-wise perception tasks, we prepare the ground truth labels for each Lidar point. The data was recorded in sequence with length varying from 15 to 30 seconds (10 Hz). To reduce the sequential dependency between frames, we down-sample the dataset by a factor of 5. The resulting dataset we use contains 2609 training frames and 996 evaluation frames, with over 20K VEHICLE and 6.7K PEDESTRIAN objects.

2) Implementation Details

All networks are trained with the same optimization settings from scratch up to 80 epochs. The ADAM optimizer is used with an initial learning rate of 0.01, a step decay of 0.1, and a batch size of 4. In order to have a fair comparison with state-of-the-art object detectors (such as PV-RCNN [17] and PointPillars [19]), which only process Lidar point clouds on the camera front-view, we extract Lidar point clouds corresponding to synchronized front-view images from the original Argoverse dataset, and train the front-view networks for most experiments. In this regard, we use the Lidar point cloud within the range length $\times$ width $\times$ height $=$ $[0, 70.4]\mathrm{m} \times [-40, 40]\mathrm{m} \times [-1.5, 4, 0]\mathrm{m}$, and do discretization at 0.1 meter voxel resolution. Besides, to employ our proposed LidarMTL network in online localization, we train a full-range network that processes Lidar point cloud within the range $[-70.4, 70.4]\mathrm{m} \times [-70.4, 70.4]\mathrm{m} \times [-1.5, 4, 0]\mathrm{m}$. All experiments are conducted using a Titan RTX GPU. The inference time for the front-view LidarMTL reaches 30FPS and for the full-range LidarMTL 7.7FPS.

B. Performance Evaluation

1) Object Detection (**OD**)

We evaluate the object detection performance using the standard Average Precision for 3D detection (AP_{3D}) and on the Bird's Eye View (BEV) (AP_{BEV}). They are measured at the Intersection Over Union IOU=0.7 threshold for "VEHICLE" objects and IOU=0.5 for "PEDESTRAIN" objects, respectively, as suggested by [34]. The IOU scores in object detection are geometric overlap ratios between bounding boxes, and indicate the localization accuracy. We report the AP scores with respect to increasing Lidar ranges ($0-30\mathrm{m}$, $30-50\mathrm{m}$, and $50-70\mathrm{m}$), as well as the mean AP scores, mAP, by averaging over all distances and object classes (similar to [35]). Besides, we report the number of trainable parameters and the inference speed for each object detectors. Tab. I summarizes the results.

The proposed multi-task network (LidarMTL) is compared against several Lidar-based object detectors. The LidarBEV network follows the same detection architecture with LidarMTL (Encoder with sparse 3D convolution + BEV detection head with 2D convolution). It serves as the baseline to study the object detection performance when introducing multiple tasks. The UNet3D network directly employs the encoder-decoder architecture from LidarMTL to predict object classes and bounding boxes on each Lidar point (without BEV detection head and pre-defined anchors),

Methods	VEHICLE		PEDESTRAIN		$mAP_{BEV}(\%)$	$mAP_{3D}(\%)$	Trainable param. (M)	Inference speed (FPS)
	$AP_{BEV}@0.7(\%)$	$AP_{3D}@0.7(\%)$	$AP_{BEV}@0.5(\%)$	$AP_{3D}@0.5(\%)$				
PV-RCNN [17]	77.5, 62.0, 21.1	63.2, 38.0, 3.8	51.8, 26.6, 4.5	45.7, 22.2, 3.0	56.1	43.2	13.10	14.6
PointPillars [19]	75.3, 57.2, 16.6	53.5, 27.8, 2.7	37.4, 22.3, 4.0	30.3, 16.8, 2.3	51.5	35.3	4.82	71.5
Second [16]	72.0, 53.9, 14.1	50.9, 25.0, 1.9	41.1, 22.8, 5.0	33.6, 17.5, 2.6	49.7	35.1	5.31	54.2
UNet3D	73.0, 35.9, 4.4	50.9, 13.7, 0.5	56.7, 25.1, 2.9	44.4, 17.4, 1.5	46.4	32.4	1.90	33.2
LidarBEV	71.8, 56.1, 14.0	50.3, 23.8, 1.7	42.0, 22.9, 4.4	36.6, 16.3, 2.1	49.9	34.6	5.31	59.6
LidarMTL	72.9, 56.9, 14.1	53.4, 24.3, 1.8	40.6, 22.9, 6.1	33.3, 17.0, 4.2	49.8	35.0	6.52	30.0

TABLE I: A comparison of Object Detection (OD) performance, as well as the number of trainable parameters and inference speed. Detections are grouped into different Lidar ranges ($0-30$m, $30-50$m, $50-70$m). The AP scores are measured at IOU=0.7 threshold for "VEHICLE" class, and IOU=0.5 for "PEDESTRAIN" class.

and is used to verify the network architecture design. Furthermore, we re-train state-of-the-art detectors, PV-RCNN [17], PointPillars [19], and Second [16], using our experimental setup. Note that PV-RCNN is a two-stage object detector, whereas all other detectors are one-stage. UNet3D directly regresses bounding box parameters, whereas the others are based on pre-defined anchors and BEV detection heads.

As Tab. I illustrates, the proposed LidarMTL network achieves similar detection accuracy to LidarBEV, SECOND, and PointPillars, with comparable number of parameters (6.52M) and reasonable inference speed (30FPS). PV-RCNN has the highest AP scores compromised by over $2\times$ more parameters and computation cost compared to LidarMTL. Though UNet3D has only 1.9M parameters, it has the worst detection accuracy with $2-3\%$ smaller *mAP* scores compared to LidarMTL, indicating the importance of adding anchor priors and BEV detection heads for precise object detection. In conclusion, the proposed LidarMTL shows competitive detection performance to other detectors regarding accuracy, model size, and inference speed.

2) Foreground (**FG**), Drivable Area (**DA**), and Ground Classification (**GC**)

We evaluate the foreground, drivable area, and ground classification tasks using the Average Precision (AP), Intersection Over Union (IOU), and classification accuracy scores at 0.5 probability threshold. Those evaluation metrics measure the classification performance at each Lidar point, and have been used as the standard metrics for road detection [36] or semantic segmentation [35]. Note that unlike the IOU metric for object detection in the previous section, here an IOU score is measured by IOU $= 100 * \text{TP}/(\text{TP+FP+FN})$ according to [35], with TP, FP, FN being the number of points categorized as true positive, false positive, and false negative samples. For each task, a task-specific UNet3D network is trained to compare with the LidarMTL network. Furthermore, since these point-wise perception tasks can be regarded as semantic segmentation, we additionally train two state-of-the-art semantic segmentation networks, namely, RangeNet++ [27] and SqueezeSegv3 [28], to perform foreground and ground classification. We do not conduct experiments for drivable area classification, because it is not mutually exclusive with other two tasks.

Experimental results are shown in Tab. II, Tab. VI, and Tab. IV. The proposed multi-task network achieves on-par performance with the single-task network, with less than 1% difference in all evaluation criterion. The network also shows

competitive results with RangeNet++ and SqueezeSegv3, verifying the effectiveness of the network architecture design.

3) Ground Height Estimation (**GH**)

We evaluate the ground height estimation performance using the Root Mean Squared Errors (RMSE) and Mean Average Errors (MAE) metrics, which are widely used in the depth prediction task for RGB camera images [37]. Tab. III reports the performance for all Lidar points, grouped with respect to the Lidar ranges ($0-30$m, $30-50$m, $50-70$m). The LidarMTL network is compared with the task-specific UNet3D network, as well as a simple heuristic which assumes a ground plane given the ego-vehicle's pose information. Note that the ground plane assumption has been widely used to remove ground Lidar points for object detection [24], [38].

Tab. III shows that the ground plane method results in larger errors at longer distances, indicating that the ground is not flat. The errors produced from the UNet3D and LidarMTL networks are much smaller than the ground plane method (over 40% RMSE reduction for all points), showing the benefits of the point-wise ground height estimation. The LidarMTL network produces slightly larger errors than the UNet3D network (< 2cm), showing small negative transfer phenomena often seen in multi-task learning [1].

A more interesting experiment is to evaluate the ground height estimation for Lidar points which belong to objects. Such information could be used to normalize objects' heights and has the potential to improve detection performance, as shown in [6]. Tab. V shows that both networks predict ground heights accurately, with RMSE errors smaller than 20cm even at $50-70$m range.

4) Intra-object part locations (**IP**)

Finally, we evaluate the performance for intra-object part location predictions, with the same evaluation metrics (RMSE and MAE) used in the previous section (Sec. IV-B.3). Both LidarMTL and UNet3D networks perform similarly, with the LidarMTL network producing slightly smaller errors (< 1) at long distance ($50-70$m) than the UNet3D network.

5) Model Size and Inference Speed

We quantitatively show the benefits of lower memory footprints and higher inference speed brought by the proposed multi-task network, compared to the "Single-task models", which performs all tasks separately by a chain of task-specific networks. In this regard, we employ the LidarBEV network introduced in IV-B.1 for object detection, and train UNet3D networks for other tasks. Starting from object detection, we gradually increase the number of perception

Methods	AP(%)	IOU (%)	Accu. (%)
RangeNet++ [27]	-	82.4	-
SqueezeSegv3 [28]	-	84.2	-
UNet3D	96.2	85.4	98.7
LidarMTL	97.0	85.6	98.7

TABLE II: Foreground (FG).

Methods	RMSE (cm)				MAE (cm)			
	All	0-30m	30-50m	50-70m	All	0-30m	30-50m	50-70m
Plane	31.2	21.0	38.0	53.5	21.6	16.3	28.2	35.5
UNet3D	17.8	7.9	21.0	40.0	7.8	4.9	9.8	20.1
LidarMTL	18.6	8.8	22.2	40.4	8.8	5.7	11.0	21.4

TABLE III: Ground heights (GH) (all Lidar points).

Methods	AP(%)	IOU (%)	Accu. (%)
RangeNet++ [27]	-	95.2	-
SqueezeSegv3 [28]	-	95.9	-
UNet3D	99.6	94.5	98.2
LidarMTL	99.6	94.0	98.0

TABLE IV: Ground areas (GC).

Methods	RMSE (cm)				MAE (cm)			
	All	0-30m	30-50m	50-70m	All	0-30m	30-50m	50-70m
Plane	20.8	17.8	27.3	35.4	15.3	12.9	22.4	28.7
UNet3D	8.6	6.5	11.7	19.1	5.5	4.4	8.0	13.9
LidarMTL	9.8	8.2	12.2	19.6	6.7	5.8	8.9	14.7

TABLE V: Ground heights (GH) (only foreground points).

Methods	AP(%)	IOU (%)	Accu. (%)
UNet3D	97.9	86.5	94.2
LidarMTL	97.4	84.5	93.4

TABLE VI: Drivable areas (DA).

Methods	RMSE				MAE			
	All	0-30m	30-50m	50-70m	All	0-30m	30-50m	50-70m
UNet3D	10.0	8.1	13.5	18.8	5.6	4.6	8.0	13.8
LidarMTL	9.9	8.2	13.3	18.1	5.7	4.7	8.0	13.2

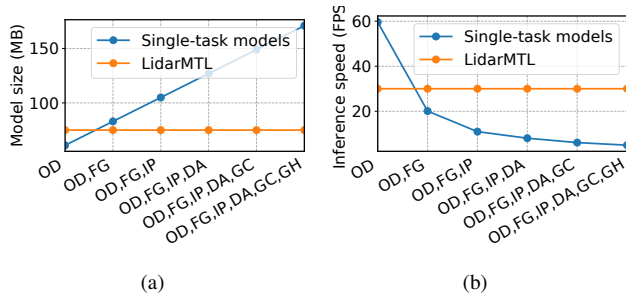TABLE VII: Intra-object part locations (IP).

(a) (b)

Fig. 3: A comparison of model size (in MegaByte) and inference speed (in FPS) required to achieve multiple perception tasks, by employing the proposed multi-task network or the "Single-task models" which performs all tasks separately by a chain of task-specific networks.

tasks, and calculate the required memory footprints and the inference speed averaged over all predictions on the evaluation data. Fig. 3(a) and Fig. 3(b) show the model size (in MegaByte) and the inference speed (in FPS), respectively. The LidarMTL network outperforms the single-task models approach, when considering more than one perception task. While the LidarMTL network remains constant model size and inference speed regardless of the number of tasks, the single-task models approach requires linearly-increasing memory and much lower inference speed. When performing all six perception tasks, the multi-task network is more than $2\times$ smaller and $6\times$ faster, showing its high efficiency, which is critical for online deployment.

C. Ablation Study

1) Number of Tasks

This section studies the performance of the single task which we focus on ("target task"), with increasing number of multiple tasks ("auxiliary tasks") in the LidarMTL network. Fig. 4(a), Fig. 4(b), and Fig. 4(c) select object detection, foreground classification, and ground height estimation as target task, respectively. We report the perception performance from the multi-task network relative to the singe-task network, with increasing number of auxiliary tasks from left to right on the x-axis. No clear tendency is observed

(a) OD + Multi-tasks (b) FG + Multi-tasks (c) GH + Multi-tasks

Fig. 4: The performance of the target task from the multi-task network trained with increasing number of auxiliary tasks, relative to the single-task network.

between the object detection performance and the number of tasks. The mAP scores fluctuate between $-1.5\% - 1\%$. Introducing more auxiliary tasks increases AP for foreground classification, as well as regression errors for ground height estimation. However, the difference is small (less than 1% AP and 1.5cm errors). In conclusion, we could achieve on-par single-task perception performance, regardless of the combination of multiple tasks.

2) Impact of Loss Weights

It is known that a proper selection of loss weight for each single task is crucial for multi-task learning [1]. In this ablation study, we train the LidarMTL network with different combinations of loss weights, and compare their multi-task performances. "Fixed (equal weights)" assumes that each loss weight is equal. "Fixed (balanced)" balances the losses to the similar scales. "Fixed (grid search)" finds a set of loss weights by grid search on the training dataset. Note that the loss weights from those three methods are fixed, and do not change during training (cf. Eq. 1). Instead, "Adaptive" employs the uncertainty weighting strategy shown by Eq. 2 to balance single-task losses adaptively. "Adaptive + grid search" first puts a set of pre-defined loss weights from the grid search, and then balances the learning with uncertainty weighting.

We report the perception performance for each single task as well as the averaged network's ranking in Tab. VIII. Surprisingly, "Fixed (balanced)" shows inferior performance even slightly worse than "Fixed (equal weights) on the

Loss weights	OD		FG	DA	GC	GH		IP		Avg. Rank
	mAP_{BEV}(%)	mAP_{3D}(%)	AP(%)	AP(%)	AP(%)	RMSE (cm)	MAE (cm)	RMSE	MAE	
Fixed (equal weights)	49.6	34.5	96.7	97.7	99.6	20.5	10.7	10.7	6.4	2.7
Fixed (balanced)	48.4	32.9	97.0	97.8	99.6	19.2	9.2	12.7	8.1	3.0
Fixed (grid search)	49.2	34.7	97.2	97.5	99.6	18.6	8.7	10.0	5.7	1.8
Adaptive [33]	49.2	34.7	97.0	97.2	99.6	24.0	14.1	10.8	6.5	3.3
Adaptive [33] + grid search	49.8	35.0	97.0	97.4	99.6	18.6	8.8	9.9	5.6	1.5

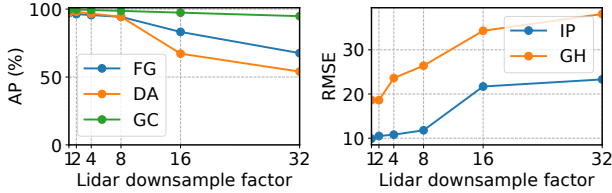TABLE VIII: A comparison among the LidarMTL networks trained with different loss weights.

Fig. 5: The performance of point-wise predictions (tasks FG, IP, DA, GC, GH) with increasingly sparse Lidar points.

Point Cloud Type		Translation RMSE (m)			Rotation RMSE (°)	Success Rate (%)
		X	Y	Z	Yaw	
Raw		1.80	1.27	1.13	1.16	83.3
GT	no DA	1.13	0.67	0.34	1.00	95.8
	no FG	1.46	0.56	0.46	0.71	95.8
	no DA, FG	1.51	0.87	0.34	1.25	95.8
LidarMTL	no DA	1.83	0.59	0.21	0.84	95.8
	no FG	1.62	0.58	0.43	0.82	95.8
	no DA, FG	1.73	0.65	0.06	1.13	100

TABLE IX: Online localization results.

averaged ranking, indicating that simply balancing losses might not be the optimal choice in multi-task learning, as different single tasks may have different learning paces. "Adaptive" ranks last, with $4 - 6cm$ larger ground height errors compared to the best results, showing the challenge to learn a proper set of loss weights from scratch. The networks trained with loss weights from grid search depict visible improvements (e.g. comparing "Fixed (grid search)" with "Fixed (equal weights)". When combining uncertainty weighting and grid search, the network slightly outperforms the "Fixed (grid search)" strategy, and achieves the best multi-task performance. We conclude the necessity of using loss weights with grid search.

3) Robustness Testing

Finally, we study the robustness of point-wise prediction tasks with increasingly sparse Lidar points. Evaluating this robustness is crucial for autonomous driving, because the sparsity of point cloud varies significantly among Lidar sensors and vehicle setup, and largely affects the perception performance [39]. In this regard, we use the LidarMTL network trained with full Lidar points (8000 voxels) to make inferences on the evaluation data with downsampled Lidar points by factors $2, 4, 8, 16, 32$. Results are shown in Fig. 5. The performance of DA, GC and IP drops slightly at downsample factors smaller than 8. FG remains high AP scores above 90% even at 32 downsample factor (i.e. 250 non-empty voxels). The performance of GH drops quickly at downsample factor 4. The experiment shows that different tasks have different robustness against Lidar point cloud sparsity.

D. Application to Online Localization

Localization in urban environments requires point cloud maps and point registration algorithms. However, Lidar-based localization typically suffered from dynamic objects and undulating road surfaces [40]. By semantically segmenting the scene, LidarMTL provides an ideal pre-processing for such localization modules. To study LidarMTL's impact on localization, we used the outputs from DA and FG to help localizing the vehicle. As a comparison, we have 4 types of

inputs for the localization algorithm: the raw scan, the point cloud without DA, the point cloud without FG, and the point cloud without both DA and FG.

We perform localization experiments on 24 trajectories spanning 2.69KM, and the vanilla NDT registration algorithm in Autoware [41] was chosen as the real-time localization module. In our experiment, the NDT voxel resolution is 1 meter. The map is created with the ground-truth scan without DA and FG downsampled to 0.2 meters.

Tab. IX shows the performance of the localization algorithm with various point cloud input. We also included the result from ground-truth DA and FG tasks for a comparison. The performance is evaluated with Root Mean Squared Error along three axis and the yaw angle. Furthermore, we listed the success rate for these tests: one is considered as a failure if the translation RMSE is larger than 3m or if the rotation RMSE is larger than $4°$. Compared with the raw input, the LidarMTL-processed inputs yield more accurate localization. Furthermore, since dynamic objects are removed from the scan, the algorithm performed robustly in complicated environments. As compared with the ground truth point cloud inputs, the LidarMTL pre-processing reaches similar level of localization accuracy and success rate. Given the stochastic nature of the NDT algorithm, the LidarMTL even outperformed ground truth segmentation in certain evaluation matrices.

V. DISCUSSION AND CONCLUSION

We have presented the multi-task network to jointly performs six perception tasks for 3D object detection and road understanding, which were only studied separately in previous works and lack quantitative analysis. Comprehensive experiments verified the network's design and the multi-task performance. The proposed multi-task network is small, fast, accurate, and useful for localization, making it highly desirable for online deployment in autonomous cars. In the future, we plan to extend the network to process multiple frames for motion estimation.

REFERENCES

[1] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *arXiv preprint arXiv:2004.13379*, 2020.

[2] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1013–1020.

[3] Y. Qian, J. M. Dolan, and M. Yang, "Dlt-net: Joint detection of drivable areas, lane lines, and traffic objects," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 21, no. 11, pp. 4670–4679, 2019.

[4] L. Chen, Z. Yang, J. Ma, and Z. Luo, "Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1283–1291.

[5] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Annual Conference on Robot Learning (CoRL)*, 2018, pp. 146–155.

[6] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7345–7353.

[7] F. Yan, K. Wang, B. Zou, L. Tang, W. Li, and C. Lv, "Lidar-based multi-task road perception network for autonomous vehicles," *IEEE Access*, vol. 8, pp. 86 753–86 764, 2020.

[8] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[9] X. Li, S. Du, G. Li, and H. Li, "Integrate point-cloud segmentation with 3d lidar scan-matching for mobile robot localization and mapping," *Sensors*, vol. 20, no. 1, p. 237, 2020.

[10] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8748–8757.

[11] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3d object detection from point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7652–7660.

[12] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526–6534.

[13] S. Shi, X. Wang, and H. Li, "Point-RCNN: 3d object proposal generation and detection from point cloud," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 770–779.

[14] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep sensor fusion for 3D bounding box estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3d object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[17] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3d object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 529–10 538.

[18] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 873–11 882.

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] L. Caltagirone, S. Scheidegger, L. Svensson, and M. Wahde, "Fast lidar-based road detection using fully convolutional neural networks," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1019–1024.

[21] R. Fan, H. Wang, P. Cai, and M. Liu, "Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 340–356.

[22] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun, "Deep multi-sensor lane detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3102–3109.

[23] J. Liang, N. Homayounfar, W.-C. Ma, S. Wang, and R. Urtasun, "Convolutional recurrent network for road boundary extraction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9512–9521.

[24] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*. Citeseer, 2015, pp. 424–432.

[25] U. Baumann, Y.-Y. Huang, C. Gläser, M. Herman, H. Banzhaf, and J. M. Zöllner, "Classifying road intersections using transfer-learning on a deep neural network," in *IEEE International Conference Intelligent Transportation System (ITSC)*. IEEE, 2018, pp. 683–690.

[26] M. Oeljeklaus, F. Hoffmann, and T. Bertram, "A combined recognition and segmentation model for urban traffic scene understanding," in *IEEE International Conference Intelligent Transportation System (ITSC)*. IEEE, 2017, pp. 1–6.

[27] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[28] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 1–19.

[29] T. Roddick and R. Cipolla, "Predicting semantic map representations from images using pyramid occupancy networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 138–11 147.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[31] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[33] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7482–7491.

[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[35] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[36] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *IEEE International Conference Intelligent Transportation System (ITSC)*, 2013.

[37] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.

[38] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8.

[39] D. Feng, Z. Wang, Y. Zhou, L. Rosenbaum, F. Timm, K. Dietmayer, M. Tomizuka, and W. Zhan, "Labels are not perfect: Inferring spatial uncertainty in object detection," *arXiv preprint arXiv:2012.12195*, 2020.

[40] W. Wen*, Y. Zhou*, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu, "Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, *Equal contribution.

[41] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, ser. ICCPS '18. IEEE Press, 2018, p. 287–296. [Online]. Available: https://doi.org/10.1109/ICCPS.2018.00035