electronics

MDPI

Article

Real-Time LiDAR Point Cloud Semantic Segmentation for Autonomous Driving

Xing Xie [1], Lin Bai [2] and Xinming Huang [2,*]

[1] School of Information Science and Technology, Nantong University, Nantong 226019, China; xiexing@ntu.edu.cn
[2] Department of Electrical and Computer Engineer, Worcester Polytechnic Institute, Worcester, MA 01609, USA; lbai2@wpi.edu
* Correspondence: xhuang@wpi.edu

Abstract: LiDAR has been widely used in autonomous driving systems to provide high-precision 3D geometric information about the vehicle's surroundings for perception, localization, and path planning. LiDAR-based point cloud semantic segmentation is an important task with a critical real-time requirement. However, most of the existing convolutional neural network (CNN) models for 3D point cloud semantic segmentation are very complex and can hardly be processed at real-time on an embedded platform. In this study, a lightweight CNN structure was proposed for projection-based LiDAR point cloud semantic segmentation with only 1.9 M parameters that gave an 87% reduction comparing to the state-of-the-art networks. When evaluated on a GPU, the processing time was 38.5 ms per frame, and it achieved a 47.9% mIoU score on Semantic-KITTI dataset. In addition, the proposed CNN is targeted on an FPGA using an NVDLA architecture, which results in a 2.74x speedup over the GPU implementation with a 46 times improvement in terms of power efficiency.

Keywords: LiDAR; point cloud; semantic segmentation; CNN; GPU; FPGA.

Citation: Xie, X.; Bai, L.; Huang, X. Real-Time LiDAR Point Cloud Semantic Segmentation for Autonomous Driving. *Electronics* **2022**, *11*, 11. https://doi.org/10.3390/electronics11010011

Academic Editors: Dong Seog Han, Kalyana C. Veluvolu and Takeo Fujii

Received: 24 November 2021
Accepted: 19 December 2021
Published: 22 December 2021

1. Introduction

Nowadays, LiDAR sensors have become indispensable for the emerging autonomous driving vehicles. LiDAR sensors are usually installed on autonomous cars for perception [1], mapping [2], and positioning [3]. Compared with a traditional camera sensor, LiDAR can capture very precise distance measurements from the surrounding environment. One of the main tasks for LiDAR-based processing on Semantic-KITTI challenges [4] is the real-time point cloud semantic segmentation.

In general, LiDAR point cloud segmentation networks can be divided into two major subcategories: the projection-based method and the point-wise method. The projection-based method is to project the 3D point cloud into a 2D spherical projection [5] or a bird's eye view (BEV) [6]. Subsequently, deep neural networks for 2D images can be directly employed. As described in [7], the point-wise method directly operates on the original 3D points. It divides the 3D space into voxel grids and utilizes neural networks to extract features from the grids. Although the number of network parameters of a point-wise method is slightly lower, the computational load is much higher owing to operations of all fully connected layers. The projection-based method can achieve comparable, state-of-the-art accuracy while running significantly faster. In this study, we followed this method based on spherical projection.

Most of the existing point cloud segmentation networks are very complex in structure and have a large number of parameters. Even when running on the latest GPUs, they can rarely match the real-time rate of a LiDAR sensor at 10 Hz. This prevents them from being applied directly to an autonomous driving system. Therefore, a lightweight CNN at a comparable accuracy is highly desirable for those time-critical applications.

In this article, we propose a real-time LiDAR point cloud semantic segmentation network, which can run in real time on the GPU. In addition, we also propose several

optimization techniques aimed at converting the ordinary CNN structure into a hardware-friendly structure. As a result, the proposed CNN network is successfully targeted on an FPGA with a processing time of only 17.7 ms. That is much faster than the 10 Hz of the LiDAR sensor data rate, leaving ample time for other perception and planning tasks for the vehicle controller. The contributions of this article are summarized as follows:

(1) To our knowledge, this is one of the first end-to-end FPGA implementations for real-time LiDAR point cloud semantic segmentation. A LiDAR sensor is directly connected to the FPGA via an Ethernet port. After pre-processing by the LiDAR driver on the embedded processor inside the FPGA, the point cloud is stored in the DDR memory that can be accessed by the on-chip CNN hardware accelerator.

(2) A real-time and lightweight CNN-named network is proposed, whose mIoU score was 47.9%on the Semantic-KITTI dataset. The network extracts features from two branches: one shallow branch for spatial information and one deep branch for context information. Its inference time on an NVIDIA RTX 3090Ti was about 38.5 ms.

(3) By balancing the on-chip memory and multipliers, our proposed network achieved a real-time processing speed of 56 frames per second (fps) when targeted on the ZCU104 MPSoC FPGA platform.

The rest of the article is organized as follows: Section 2 summarizes the existing research on LiDAR point cloud semantic segmentation and the FPGA implementations of segmentation networks. In Section 3, the proposed segmentation network model is described in detail with evaluation performance on the GPU. The FPGA system architecture and its implementation results are discussed in Sections 4 and 5, respectively. Finally, Section 6 concludes the entire article.

2. Related Work

2.1. Semantic Segmentation of Lidar Point Clouds

In recent years, the use of deep neural networks for semantic segmentation of 3D LiDAR point clouds has made great progress [7–12].These mainstream network structures for semantic segmentation basically use full convolutional networks [13], multi-branch models [14], and encoder–decoder structures [2]. However, these network structures are difficult to achieve the balance of processing speed and segmentation accuracy. Based on the above reasons, we propose a multi-branch network model to extract spatial features and context features separately, and then these features are fused together to generate the semantic information, thus achieving both a real-time processing speed and a high precision of segmentation.

The core difference between these advanced methods is not only in network design but also in the representation of point-cloud data. Point-wise methods directly process the original irregular 3D points without any pre-processing, such as the mainstream method, namely, PointNet [15–18] and its subsequent version PointNet++ [16]. Although such methods are powerful on small point clouds, the computation load on larger point-cloud data sets becomes very heavy and requires a much longer processing time.

The projection-based method is to project the 3D point cloud into a 2D bird's eye view (BEV) [6] or spherical projection [5]. PolarNet [19] used the BEV projection, which projects the point cloud data into the BEV representation in polar coordinates. SqueezeSeg [8], SqueezeSegV2 [10], SqueezeSegV3 [20], and RangeNet++ [9] utilized the spherical projection mechanism to convert a 3D point cloud into a frontal-view image or a spherical-projection image and adopted standard 2D convolutional networks in image space for semantic segmentation. SalsaNext [7] made a series of improvements to the backbone network in SalsaNet [21], adding a new global context block and an improved encoder–decoder [22] to achieve state-of-the-art results in 3D LiDAR semantic segmentation using a spherical-projection image as input.

2.2. Fpga Implementations of Segmentation Networks

In autonomous driving or advanced driver-assistance systems (ADAS), the LiDAR point-cloud-segmentation algorithm must fulfill a real-time requirement, so it is often

implemented on embedded platforms such as ASIC, FPGA, or mobile CPU/GPU processors [23,24]. Some of the previous works [25,26] mainly used a one-dimensional shrinking array to accelerate matrix multiplication on FPGA, which achieved an efficient resource utilization and a low bandwidth. The newly proposed neural network architecture [27] used the single-instruction multiple-data (SIMD) structure for matrix multiplication.Continental AG released the assisted autonomous driving control unit (ADCU) based on the Zynq UltraScale+ MPSoC chip. The control unit supported LiDAR processing but did not reveal technical details. In [28,29], the LiDAR sensor was connected to the PC via Ethernet. After pre-processing the LiDAR point cloud on the PC, feature maps were fed to the neural-network accelerator in the FPGA. NVIDIA proposed the DRIVE AGX autonomous driving computer platform based on the Xavier SoC chip, which can process the point cloud data received from the LiDAR sensor.

3. Proposed Network

In this section, we describe our method in detail from the point-cloud representation, the CNN architecture, and the network training. Finally, the performance evaluation of the network is given.

3.1. Spherical Projection of LiDAR Point Cloud

As in [9], we projected a sparse 3D LiDAR point cloud onto a spherical surface, and the generated range view (RV) image of the LiDAR allows standard 2D convolution to be processed.

In the RV image, each raw LiDAR point (x, y, z) is mapped to the image coordinate (u, v) as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - \arctan(y, x)\pi^{-1}\right]w \\ \left[1 - \left(\arcsin\left(z, r^{-1}\right) + f_{down}\right)f^{-1}\right]h \end{pmatrix} \quad (1)$$

where (u, v) are the image coordinates, (h, w) are the height and width of the desired range image representation, r represents the range of each point as $r = \sqrt{x^2 + y^2 + z^2}$, and f defines the sensor vertical field of view as $f = |f_{\text{dowm}}| + |f_{up}|$.

For each point projected to (u, v), we used its 3D point coordinates (x, y, z), distance index r, and intensity value i as the characteristics and stacked them along the channel dimension. In this way, we can represent a 3D LiDAR point cloud as an RV image with the shape $[w \times h \times 5]$ and feed it to the network, and then the point-cloud segmentation can be transformed to image segmentation.

3.2. Network Architecture

Our proposed network follows a multi-branch model design, extracting the spatial and contextual features separately and then fusing these features to recover the network information. The network structure is shown in Figure 1. The network is inspired by ContextNet [30], BiSeNet [31], and SalsaNext [7]. The RV image projection of the point cloud is used as the input of the network, as described in Section 3.1.

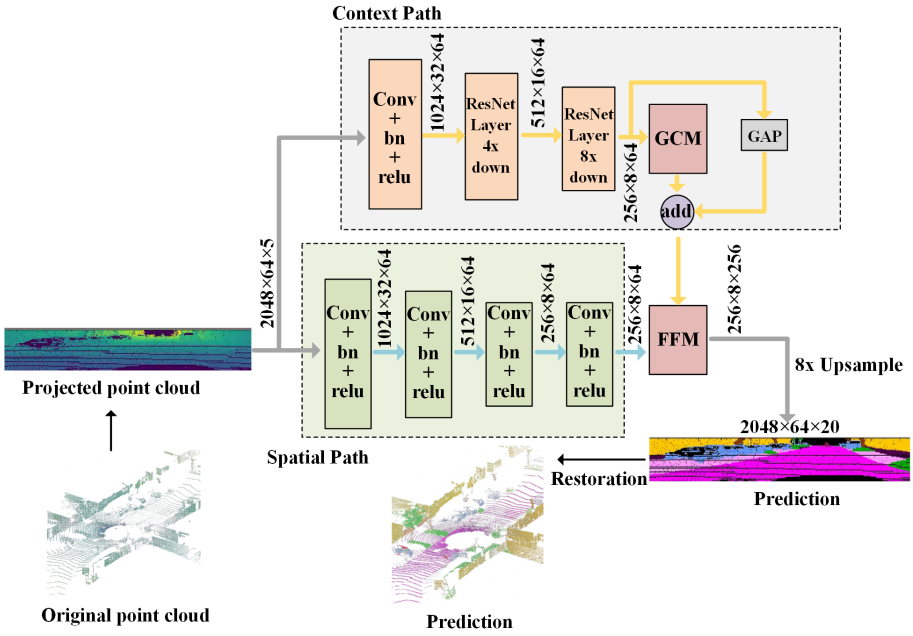


Figure 1. Architecture of the proposed CNN for point-cloud semantic segmentation.

Context path: Semantic segmentation involves pixel-level classification and spatial location information of pixel categories. Therefore, effective context information and original image spatial detail information are important to semantic-segmentation tasks. In order to capture the contextual information of different global regions, we placed an input convolutional layer and two residual modules from ResNet18 [32] at the front end of the network for eight times fast down-sampling. We added global average pooling (GAP) at the end of ResNet18, thus providing a larger receptive field. Next, a global context module (GCM) was introduced to refine the context information, and we used it to guide the feature learning of the current path. In our previous work [23], GCM was modified from the attention refinement module as in [31]. As shown in Figure 2a, a GCM consists of a global-average-pooling layer and a convolutional layer that extracts global context features. These improved global features are applied to contextual feature fusion by multiplication. The sigmoid layer determines whether to apply global features or not. It integrates the global context information easily without any up-sampling operation. Therefore, the computation cost is negligible.

Spatial path: The spatial path mainly captures the spatial details of the input image and contains only four convolutional layers. The first three convolutional layers each contain a stride 2 convolution, followed by batch normalization [33] and ReLU [34]. Therefore, the output feature map extracted by this path was 1/8 of the original image. Due to the large spatial size of the feature map, it encodes rich spatial information. Figure 1 shows the details of the structure.

The feature-fusion module (FFM) [31] is to fuse the output features from the context path and the spatial path, as shown in Figure 2b. Since the features of the two paths are not the same, the two parts of the features cannot be simply weighted, but they are superimposed through concatenation. The FFM module includes the global-average-pooling layer, 11 convolutional layers, an ReLU activation layer, and a sigmoid layer. At the end of the network, the FFM output is up-sampled eight times using the bi-linear resize algorithm to produce the output in the same size of the input.

The number of channels is chosen to be a factor of 32. This is based on the number of parallelism that the hardware accelerator can support in order to maximize its efficiency.

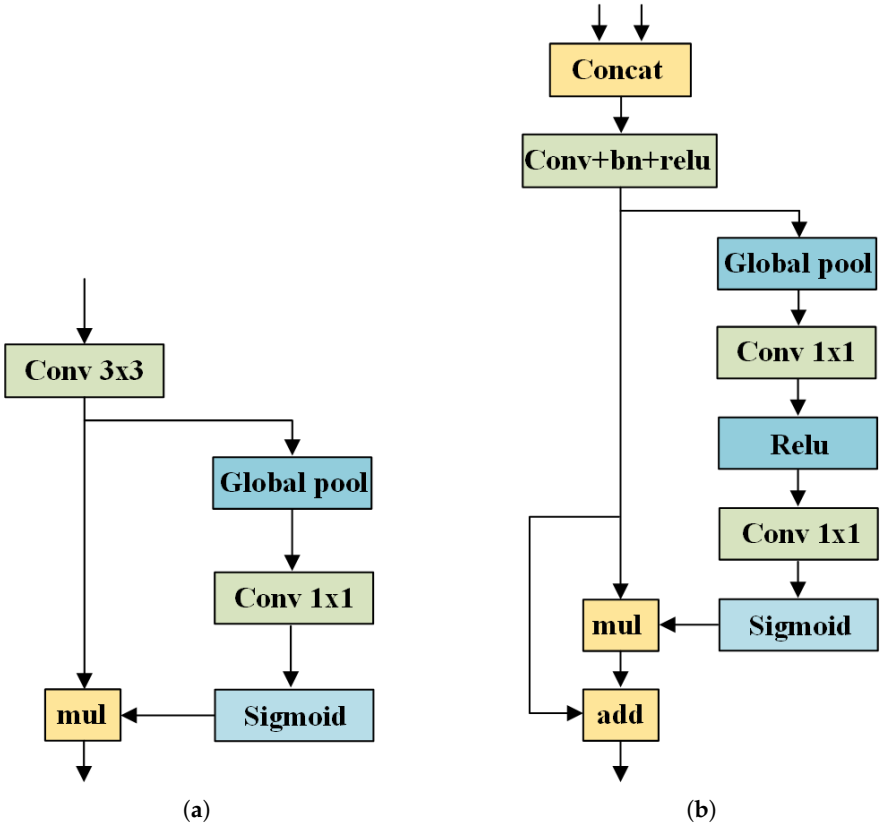


Figure 2. (**a**) Structure of GCM; (**b**) structure of FFM.

3.3. Training Details

This 3D point-cloud semantic-segmentation network was implemented using PyTorch, and it was trained on a single NVIDIA RTX 3090Ti GPU. Following previous work [9,10], we performed spherical projection processing on all points following Equation (1). We projected all points in a scan into a 64×2048 image. If multiple points are projected to the same pixel of a 2D image, the point with the largest distance is retained. Then, we used 2D convolution to process the range-view image to obtain a 2D predicted label map, and then we restored it to 3D space.

During training, the network was trained with an initial learning rate of 0.01, and the batch size was set to 24. In the inference process, the original point cloud via spherical projection was fed into the network, and the 2D prediction result was obtained. Then, we used the restoration operation to obtain the 3D prediction, as in the previous work [9,10].

3.4. Dataset and Evaluation

Semantic-KITTI [4] is a large-scale dataset for 3D LiDAR point-cloud segmentation, including semantic segmentation and panoptic segmentation. The dataset contains 22 sequences of point-cloud data. We follow the same protocol in [9], where the sequences between 00–10 are the training data, and the sequence 08 is used for validation. The remaining sequences between 11–21 are testing data.

Evaluation Metric: In order to evaluate the proposed method, we followed the official guidance and used the mean intersection-over-union (mIoU) as the evaluation metric defined in [4,35], which can be formulated as:

$$mIOU = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c + FN_c} \tag{2}$$

where TP_c, FP_c, and FN_c correspond to the number of true-positive, false-positive, and false-negative predictions for class c where c is the class number.

Quantitative Results:Following previous work [9], we used mIoU over 19 categories to evaluate the accuracy. Table 1 shows the quantitative results obtained compared with other state-of-the-art point-wise and projection-based methods. The mean IoU score (47.9%) of our proposed model was slightly lower than that of the state-of-the-art models due to the reduced network complexity. In contrast to the baseline BiseNet, we obtained an improvement greater than 6.5% in terms of the accuracy, with the best performance in 18 of the 19 categories. When it comes to the performance of each individual category, the proposed network has comparable performance similar to other approaches.

For qualitative evaluation, Figures 3 and 4 show some semantic-segmentation results generated by the 3D point-cloud segmentation network on the Semantic-KITTI test set. In order to compare the results more easily, Figure 3 shows the results using spherical projection, with each color representing a different semantic class. We can see that ground points are divided into a road and a sidewalk. In particular, the road needs a significant amount of contextual information and information from neighboring points, since a small curb usually distinguishes the sidewalk from the road. The network clearly can distinguish the objects such as cars on the road.

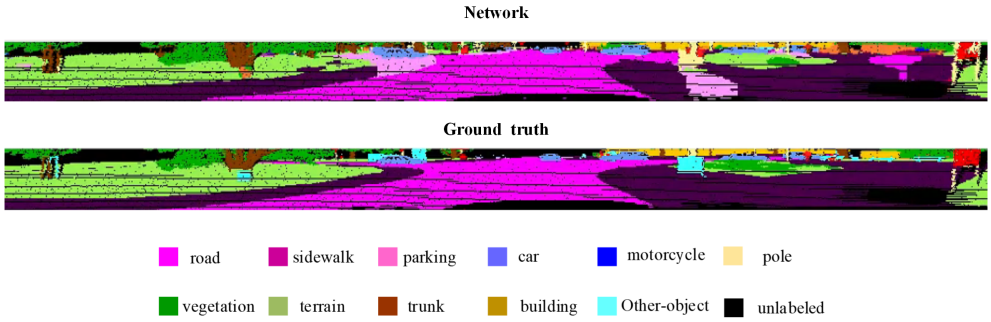
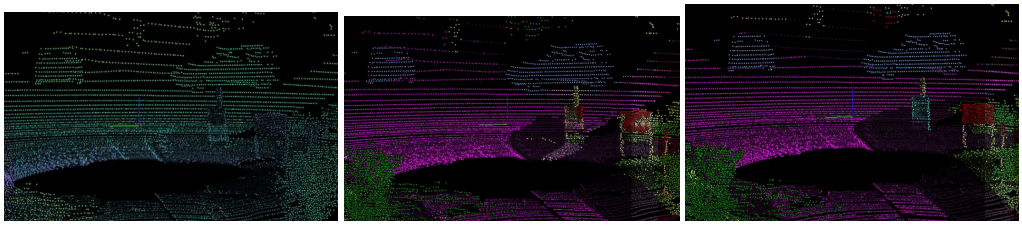


Figure 3. An inference result of the proposed segmentation network.



(**a**) (**b**) (**c**)

Figure 4. Qualitative results of our semantic segmentation network. (**a**) Input volume; (**b**) segmentation network predictions; (**c**) ground truth.

Table 1. Segmentation mIoU (%) results on the Semantic-KITTI test dataset.

Methods	mIoU	Car	Bicycle	Motorcycle	Truck	Other-Vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-Ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-Sign
PNet [15]	14.6	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0	61.6	15.8	35.7	1.4	41.4	12.9	31	4.6	17.6	2.4	3.7
PNet++ [16]	20.1	53.7	1.9	0.2	0.9	0.2	0.9	1	0	72	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30	6	8.9
SPGraph [12]	20	68.3	0.9	4.5	0.9	0.8	1	6	0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17	18.3	10.5
SPLATNet [36]	22.8	66.6	0	0	0	0	0	0	0	70.4	0.8	41.5	0	68.7	27.8	72.3	35.9	35.8	13.8	0
TgConv [37]	35.9	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9	82.8	44.2	75.5	42.5	55.5	30.2	22.2
RLNet [38]	50.3	94	19.8	21.4	42.7	38.7	47.5	48.8	4.6	90.4	56.9	67.9	15.5	81.1	49.7	78.3	60.3	59	44.2	38.1
SqueezeSeg [8]	29.5	68.8	16	4.1	3.3	3.6	12.9	13.1	0.9	85.4	26.9	54.3	4.5	57.4	29	60	24.3	53.7	17.5	24.5
SSG-CRF [8]	30.8	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3
SSGV2 [10]	39.7	81.8	18.5	17.9	13.4	14	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3
SSGV2-CRF [10]	39.6	82.7	21	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41	68.5	36.9	58.9	12.9	41
SalsaNet [21]	45.4	87.5	26.2	24.6	24	17.5	33.2	31.1	8.4	89.7	51.7	70.7	19.7	82.8	48	73	40	61.7	31.3	41.9
RangeNet21 [9]	47.4	85.4	26.2	26.5	18.6	15.6	31.8	33.6	4	91.4	57	74	26.4	81.9	52.3	77.6	48.4	63.6	36	50
RangeNet53 [9]	49.9	86.4	24.5	32.7	25.5	22.6	36.2	33.6	4.7	91.8	64.8	74.6	27.9	84.1	55	78.3	50.1	64	38.9	52.2
RangeNet53* [9]	52.2	91.4	25.7	34.4	25.7	23	38.3	38.8	4.8	91.8	65	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
SSGV3 [20]	55.9	92.5	38.7	42.1	29.6	33	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89	59.4	82	58.7	65.4	49.6	58.9
SalsaNext [7]	59.5	91.9	48.3	38.6	38.9	31.9	60.2	59	19.4	91.7	63.7	75.8	29.1	90.2	64.2	81.8	63.6	66.5	54.3	62.1
BiseNet [31]	41.4	85.9	10.7	6.2	38.7	11	15.9	49.2	0	90.9	30.1	74.8	0	75.2	29.2	79.5	47.5	72	40.4	29.6
Ours	47.9	87.3	19.4	27.6	46.6	28.3	43.5	57.3	1.6	91.2	34.1	75.2	2.1	81.3	36.9	78	49.4	73.1	41.5	36.3

3.5. Run-Time Evaluation on Gpu

In autonomous driving, the processing time of the network must meet real-time requirements. In order to obtain fair statistics, all measurements were performed on the same single NVIDIA RTX 3090Ti-24GB card using the entire Semantic-KITTI dataset. The total run-time performance of our proposed network compared with other networks is shown in Table 2. As depicted in Table 2, compared with BiseNet [31], our method clearly shows better performance, while the parameters were reduced by about seven times. When the uncertainty calculation is excluded and a fair comparison is made with the deterministic model, our model can run at 26 Hz. Note that the high speed we achieved is significantly faster than the frame rate of mainstream LiDAR sensors that is are at 10 Hz or 20 Hz [39].

Table 2. Run-time performance on the SemanticKITTI test set.

	Processing Time (msec)	Speed (fps)	Parameters
BiSeNet [31]	59.60 ms	17 Hz	14.1 M
Ours (GPU)	38.50 ms	26 Hz	1.9 M
Ours (FPGA)	17.76 ms	56 Hz	1.9 M

4. FPGA-Based System Architecture

The FPGA-based system architecture for the proposed LiDAR point-cloud semantic-segmentation network is shown in Figure 1, which is partitioned into the software part on the ARM processors and the hardware part on the programmable logic (PL). The image adjustment of the input and output of the neural network is mainly processed using the OpenCV library functions [40] running on the ARM processor. In this study, we implemented the NVIDIA Deep Learning Accelerator (NVDLA) framework on the PL side that is interconnected with the dual-core ARM processor through the AXI4 bus. The CNN inference executes on the NVDLA architecture as an FPGA-based hardware accelerator.

The NVDLA hardware-accelerator architecture consists of the following four components: (1) calculation of the convolution engine unit; (2) the activation engine; (3) the pooling engine; (4) the feature-map (FM) buffer and the weight buffer. The 2D convolution and adder tree were embedded in the convolution engine module. The above parts were configured based on the on-chip resources available on the target FPGA platform, and the block diagram of the hardware architecture is illustrated in Figure 5.

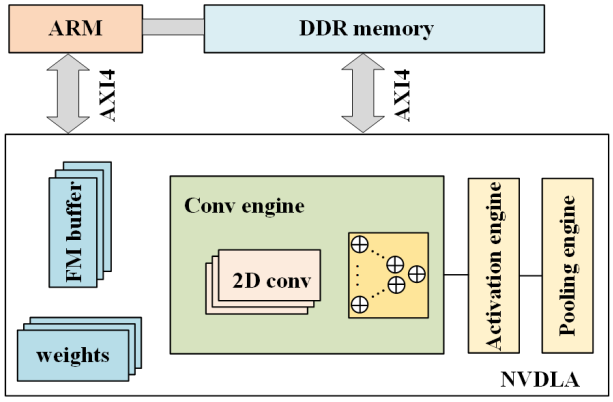


Figure 5. Hardware architecture of the CNN accelerator on the FPGA.

4.1. Software Tasks on Arm Processor

As shown in Figure 6, in order to fully use the computing resources available on the FPGA, our point-cloud processing algorithm was partitioned to the ARM processor and programmable logic. The neural network input pre-processing and output-image resize were processed on the ARM processor as software, while the entire CNN inference was implemented using NVDLA on the programmable logic. All three tasks were scheduled as a pipeline, thus increasing system throughput.

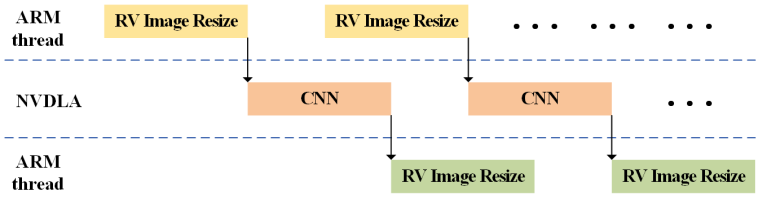


Figure 6. Pipeline processing between the ARM processor and the CNN accelerator.

4.2. Convolution Engine

As shown in Figure 7, the NVDLA standard convolution engine was configured with a convolution buffer. When the convolution operation is enabled, the addresses of FM and weights in the main memory are configured in the registers, and then data are pre-fetched to the FM buffer for operations, reducing the overhead of memory access. The core of the convolution engine is the multiply-accumulate (MAC) array. The array size is configurable by optimizing it for parallel operations. In Figure 7, taking a 3×3 kernel as an example, it consists of nine multipliers and an adder tree. In this work, the size of the MAC array was 32×32.

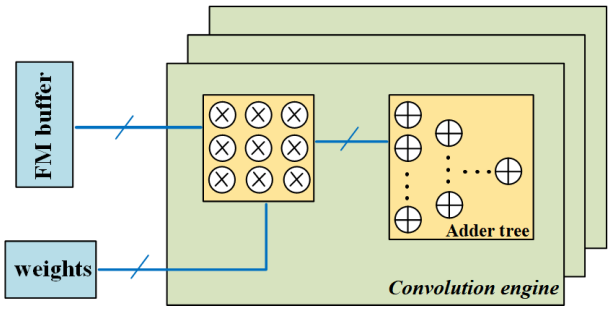


Figure 7. Hardware structure of the convolution engine.

4.3. GCM Module and FFM Module

Both GCM and FFM modules need to operate using completely different computing modes. The pooling engine of NVDLA was adopted to realize the global average pool, which was used to calculate the average value of the whole channel. The following 1×1 convolution was mathematically equivalent to a vector-matrix multiplication and can be routed to the convolution-engine module. The NVDLA activation engine module supports a wide variety of linear and non-linear operations, provides native support for linear functions, and uses look-up tables (LUTs) to implement non-linear functions. Therefore, the calculation of the ReLU function and the sigmoid function can be realized.

4.4. Memory Mapping

The on-chip memory separately stores weight-data, feature-mapping, and global-pooling results. In order to increase the processing speed and to reduce the data-transfer rate between FPGA and DDR memory, NVDLA uses a ping-pong buffer mechanism to improve system efficiency. The use of two register banks can reduce the reprogramming delay. In this method, the second group of buffers were programmed at the same time when the first group of buffers is processing the convolution calculation.

4.5. Quantization

To maximize the computation capability of NVDLA, fixed-point operations are preferred. In this study, we choose the eight-bit integer quantization for NVDLA hardware implementation. Firstly, from a storage perspective, the memory space for eight-bit weights is only half of that for 16-bit quantization. Secondly, from a hardware-resources perspective,

each DSP slice inside the FPGA can perform two eight-bit multiplications concurrently but can only perform one 16-bit multiplication. This results in faster computations.

5. Results and Discussion

The target hardware platform was a Zynq UltraScale+ MPSoC ZCU104 development board. The PetaLinux operating system was running on the ARM core. The test setup is shown in Figure 8, where the ZCU104 board and LiDAR were connected through the Ethernet port via a UDP protocol. After being processed by the Velodyne LiDAR driver, users can send Python commands to the ARM processor within the FPGA chip (PS side), which receives point-cloud data from LiDAR and stores them in the DDR memory after pre-processing. During execution, the CNN accelerator reads point-cloud data from the DDR or stores the intermediate data to DDR using DMA via a high-performance (HP) port according to the on-chip AXI bus protocol.

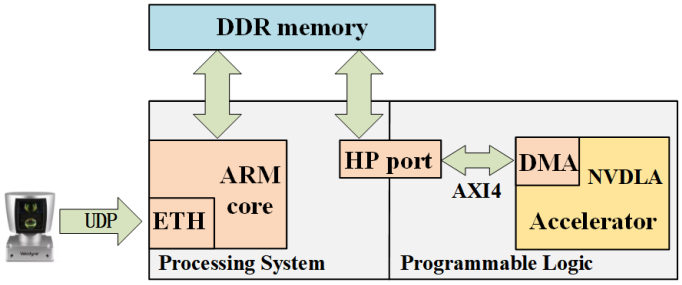


Figure 8. Overall system setup with LiDAR connected to the FPGA board directly.

When running at 250 MHz, this CNN accelerator can process one frame of a point cloud within 17.76 ms. As mentioned earlier, there are few existing FPGA implementations of 3D point-cloud segmentation using CNN in the literature. Performance and efficiency comparisons with similar works on FPGAs are not yet available.

The hardware resource consumption is summarized in Table 3. The bottleneck of this design is DSP resources, since 95.78% of those are already in use. Increasing the parallelism would require more DSP slices, and a larger FPGA would be necessary. Furthermore, due to the large feature map size, 100% on-chip memory, including both BRAM and URAM, were utilized to buffer as many feature maps or parameters as possible.

Table 3. FPGA Resource utilization for the CNN accelerator.

FPGA Resource	Used	Available	Utilization
LUT	202,679	230,400	87.97%
FF	227,220	460,800	49.31%
DSP	1655	1728	95.78%
BRAM	83	312	26.60%
URAM	96	96	100%

The FPGA performance on the Semantic-KITTI dataset is shown in Table 4. After replacing all the large kernel convolutions with uniform kernel size and quantization using INT8 fixed-point weights, the mIoU of network on FPGA was 46.4%, which is slightly less than the floating-point computations on the GPU. The estimated power consumption of the FPGA implementation is shown in Table 5. For comparison purposes, the GPU power was estimated at 115 Watts. Considering the FPGA processing time was 2.17 times faster than the GPU, our FPGA implementation was 46 times better than the GPU in terms of power efficiency.

Table 4. Run-time Performance of the Proposed Approach on Semantic-KITT Dataset.

Device	Precision	mIoU	Processing Time
GPU	Float32	47.9%	38.50 ms
FPGA	Int8	46.4%	17.76 ms

Table 5. Power Consumption of the FPGA Design.

Power	Power Consumed
Dynamic	4.719 W
Static	0.716 W
Total	5.435 W

6. Conclusions

In this study, we proposed a lightweight CNN for the task of LiDAR point-cloud semantic segmentation. For a feature map size of 64 × 2048, this network achieved a 47.9% mIoU score on the Semantic-KITTI dataset with a processing time of 38.5 ms on the RTX 3090Ti GPU. When comparing to state-of-the-art networks, the network achieved similar error performance but only used 13% of the parameters. Furthermore, the proposed network was successfully targeted on an MPSoC FPGA platform using an NVDLA hardware architecture, which speeds up the processing time by a factor of 2.17 compared to its GPU implementation.

Author Contributions: Conceptualization, X.X. and L.B.; methodology, X.X., L.B.; software, X.X.; formal analysis, X.X., L.B.; data curation, L.B. and X.X.; writing—original draft preparation, X.X; writing—review and editing, X.H.; supervision, X.X.; project administration, X.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yin, Z.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4490–4499.
2. Zhang, C.; Luo, W.; Urtasun, R. Efficient Convolutions for Real-Time Semantic Segmentation of 3D Point Clouds. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; pp. 399–408.
3. Yin, H.; Wang, Y.; Ding, X.; Tang, L.; Xiong, R. 3D LiDAR-Based Global Localization Using Siamese Neural Network. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1380–1392. [CrossRef]
4. Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; Gall, J. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27–28 October 2019.
5. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-View 3D Object Detection Network for Autonomous Driving. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6526–6534.
6. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S. Joint 3D Proposal Generation and Object Detection from View Aggregation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–8.
7. Cortinhal, T.; Tzelepis, G.; Aksoy, E.E. SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving. *arXiv* **2020**, arXiv:2003.03653.
8. Wu, B.; Wan, A.; Yue, X.; Keutzer, K. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 1887–1893.
9. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Venetian Macao, Macau, 3–8 November 2019; pp. 4213–4220
10. Wu, B.; Zhou, X.; Zhao, S.; Yue, X.; Keutzer, K. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 4376–4382.

11. Yuan, W.; Shi, T.; Peng, Y.; Lei, T.; Ming, L. PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud. *arXiv* **2018**, arXiv:1807.06288.
12. Landrieu, L.; Simonovsky, M. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4558–4567.
13. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 640–651. [CrossRef] [PubMed]
14. Poudel, R.; Liwicki, S.; Cipolla, R. Fast-SCNN: Fast Semantic Segmentation Network. *arXiv* **2019**, arXiv:1902.04502.
15. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 77–85.
16. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv* **2017**, arXiv:1706.02413.
17. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.; Bronstein, M.; Solomon, J. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* **2019**, 38, 1–12. [CrossRef]
18. Klokov, R.; Lempitsky, V. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 863–872.
19. Zhang, Y.; Zhou, Z.; David, P.; Yue, X.; Foroosh, H. PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 9598–9607.
20. Xu, C.; Wu, B.; Wang, Z.; Zhan, W.; Vajda, P.; Keutzer, K.; Tomizuka, M. *SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation*; Springer: Berlin/Heidelberg, Germany, 2020.
21. Aksoy, E.E.; Baci, S.; Cavdar, S. SalsaNet: Fast Road and Vehicle Segmentation in LiDAR Point Clouds for Autonomous Driving. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 926–932.
22. Berman, M.; Triki, A.R.; Blaschko, M.B. The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4413–4421.
23. Bai, L.; Lyu, Y.; Huang, X. RoadNet-RT: High Throughput CNN Architecture and SoC Design for Real-Time Road Segmentation. *IEEE Trans. Circ. Syst. Regul. Pap.* **2020**, 68, 704–714. [CrossRef]
24. Bai, L.; Zhao, Y.; Elhousni, M.; Huang, X. DepthNet: Real-Time LiDAR Point Cloud Depth Completion for Autonomous Vehicles. *IEEE Access* **2020**, *8*, 7825–227833. [CrossRef]
25. Shen, J.; You, H.; Qiao, Y.; Mei, W.; Zhang, C. Towards a Multi-array Architecture for Accelerating Large-scale Matrix Multiplication on FPGAs. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
26. Wu, G.; Yong, D.; Miao, W. High performance and memory efficient implementation of matrix multiplication on FPGAs. In Proceedings of the International Conference on Field-Programmable Technology, FPT 2010, Beijing, China, 8–10 December 2010.
27. Chang, J.W.; Kang, S.J. Optimizing FPGA-based convolutional neural networks accelerator for image super-resolution. In Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), Jeju, Korea, 22–25 January 2018; pp. 343–348.
28. Lyu, Y.; Bai, L.; Huang, X. Real-Time Road Segmentation Using LiDAR Data Processing on an FPGA. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
29. Lyu, Y.; Bai, L.; Huang, X. ChipNet: Real-Time LiDAR Processing for Drivable Region Segmentation on an FPGA. *IEEE Trans. Circ. Syst. Regul. Pap.* **2019**, *66*, 1769–1779. [CrossRef]
30. Poudel, R.; Bonde, U.; Liwicki, S.; Zach, C. ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-time. *arXiv* **2018**, arXiv:1805.04554.
31. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. In Proceedings of the 15th European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778
33. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
34. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), Ft. Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
35. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Beijbom, O. nuScenes: A multimodal dataset for autonomous driving. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11618–11628.
36. Hang, S.; Jampani, V.; Sun, D.; Maji, S.; Kautz, J. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 2530–2539.

37. Tatarchenko, M.; Park, J.; Koltun, V.; Zhou, Q.Y. Tangent Convolutions for Dense Prediction in 3D. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
38. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 11105–11114.
39. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the 2012 IEEE conference on computer vision and pattern recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
40. The OpenCV Library. Available online: https://github.com/opencv/opencv/wiki/CiteOpenCV (accessed on 16 September 2021).