RoadNet-RT: High Throughput CNN Architecture and SoC Design for Real-Time Road Segmentation

Lin Bai, *Graduate Student Member, IEEE*, Yecheng Lyu, *Graduate Student Member, IEEE*, and Xinming Huang, *Senior Member, IEEE*

***Abstract*—In recent years, convolutional neural network (CNN) has gained popularity in many engineering applications especially for computer vision. In order to achieve better performance, more complex structures and advanced operations are incorporated into neural networks, which results in very long inference time. For time-critical tasks such as autonomous driving and virtual reality, real-time processing is fundamental. In order to reach real-time processing speed, a lightweight, high-throughput CNN architecture namely RoadNet-RT is proposed for road segmentation in this article. It achieves 92.55% MaxF score on KITTI road segmentation dataset. The inference time is about 9 ms per frame when running on GTX 1080 GPU. Comparing to the state-of-the-art network, RoadNet-RT speeds up the inference time by a factor of 17.8 at the cost of only 3.75% loss in accuracy. What is more, on CamVid dataset its accuracy is 92.98%. Several techniques such as depthwise separable convolution and non-uniformed kernel size convolution are optimized in the hardware accelerator design. The proposed CNN architecture has been successfully implemented on a ZCU102 MPSoC FPGA that achieves the computation capability of 331 GOPS using INT8 quantization. The system throughput reaches 196.7 frames per second with input image size of 280 × 960. The source code is published at https://github.com/linbaiwpi/RoadNet-RT.**

***Index Terms*—Road segmentation, real-time, FPGA, neural network.**

I. Introduction

NOWADAYS autonomous vehicles have become one of the most promising technologies. Owing to the continuous development of Convolutional Neural Networks (CNNs), many recent research were focused on improving the accuracy performance of the perception system for autonomous vehicles, such as vehicles or pedestrians detection [1], [2], depth completion [3], road segmentation [4], [5] and object tracking [6]. However, most of these neural networks are very deep with a huge number of parameters. Even running on a state-of-the-art GPU, few of them are able to process sensor data in real-time. This prevents them being applied to time-critical tasks such as autonomous driving. Therefore, a fast lightweight CNN with reasonable accuracy is valuable to those time-critical applications.

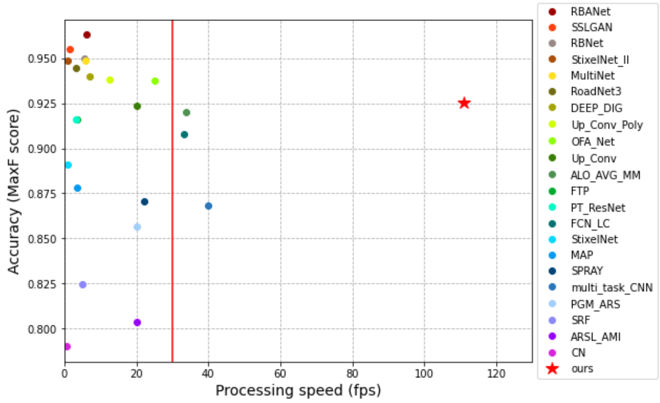


Fig. 1. Processing speed v.s. accuracy on the KITTI road segmentation test dataset. Red star indicates our method, and colored dots represent other methods. All of these solutions are tested on GPU/CPU which listed in KITTI leaderboard of road segmentation task. Red line is the border of real-time.

Road segmentation is one of the fundamental perception tasks for autonomous driving, which tells the vehicles where the drivable region is. This task has been well studied by many researchers concerning to the accuracy performance measured by benchmarks. While as a time-critical task, only 3 of the existed methods are able to process in real-time as illustrated in Fig. 1, where the red line indicates the real-time processing speed at 30 frames per second (fps), and none of their throughput exceeds 40 fps. As a fundamental task prior to path planning and dynamic control, road segmentation is expected to process input images at a much faster frame rate, such that it guarantees the real-time response of an autonomous driving system. Thus, there is an urgent need of real-time road segmentation that can process each image within a very short time while maintaining good accuracy, which bridges the gap between academic research and industry practice.

In this article, we propose RoadNet-RT, a real-time road segmentation network, which is able to run in real-time on a GPU. Besides, we have summarized some optimization

Manuscript received June 9, 2020; revised August 26, 2020 and October 16, 2020; accepted November 4, 2020. This work was supported in part by the U.S. NSF under Grant CCF-2006738 and in part by The MathWorks Fellowship. This article was recommended by Associate Editor S. Yin. *(Corresponding author: Xinming Huang.)*

The authors are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA (e-mail: xhuang@wpi.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2020.3038139.

Digital Object Identifier 10.1109/TCSI.2020.3038139

techniques aiming to convert ordinary CNN structures into hardware friendly ones. As a demonstration, RoadNet-RT has been successfully implemented on an FPGA by applying these techniques, resulting real-time processing on hardware. The contributions of this article are summarized as following:

- A lightweight high throughput CNN named RoadNet-RT is proposed, whose segmentation accuracy is 92.55% on KITTI road segmentation leaderboard. RoadNet-RT extracts features from two branches, one shallow branch for spatial information and one deep branch for context information. Its inference time on NVIDIA GTX 1080 is about 9 ms. When comparing to the state-of-the-art RBANet [4], this network reduces the inference time by 94.4%, with only 3.75% loss in accuracy.
- Aimed at providing the general guidelines on how to transform a segmentation CNN into a hardware friendly one with both computation and bandwidth efficiencies, we investigate several hardware optimization techniques through a series of experiments with quantitative results. For instance, how to employ depthwise separable convolution, how to deal with convolutions with different kernel size and dilated convolution, and whether using batch normalization are studied.
- An efficient hardware accelerator has been implemented on a ZCU102 MPSoC FPGA platform. By balancing the bandwidth and computation capability, this accelerator can process 196.7 image frames per second with INT8 quantization, equivalent to the efficiency of 331 Giga Operations Per Second (GOPS).

The rest of the paper is organized as following: Sec. II summarizes the existing research on road segmentation, real-time segmentation CNNs and the FPGA implementations of segmentation networks. In Sec. III, the proposed segmentation network model is described together with its training details. An in-depth study of network optimization techniques for hardware efficiency and accuracy performance is presented in Sec. IV. The FPGA implementation and its results are discussed in Section V and VI, respectively. Sec. VII concludes the entire paper.

II. Related Work

A. Road Segmentation

Lots of research efforts have been paid on road segmentation task in KITTI. The RBANet proposed in [4] adopted the classical encoder-decoder structure. Instead of using the direct skip connection in U-Net [7] and SegNet [8], a residual refinement module bridged encoder and decoder parts, which consisted of reversed attention and boundary attention mechanisms. So that high-resolution spatial details were preserved for decoding. Atrous Spatial Pyramid Pooling (ASPP) module was also utilized in RBANet. For images size 360 × 720 running on GTX Titan XP, the processing time was 0.16 second per frame. In [9], SSLGAN served to train unlabeled data and enhanced road feature representations using a discriminator from GAN. Labeled data contain many redundant areas, so training both labeled and unlabeled data prevents the overfitting problem and accelerates the convergence speed. Its processing speed was 0.7s per frame on TITAN X. A road and road boundary detection network (RBNet) was proposed in [5]. Based on a Bayesian network, RBNet could simultaneously estimate the probabilities of a pixel on the image belonging to the road and road boundary so that the road and road boundary detection were combined into a single process. It was able to process each frame in 0.18s on Tesla K20c (5 GB). StixelNet [10] posed generic static obstacles represented as stixels and learnt directly using a CNN. StixelNet II [10] was a unified network with real-time detection capability for both categorized and un-categorized objects. This network performed well on column-based obstacle detection and road segmentation but was not sensitive to the distinction of road boundaries. MultiNet [11] utilized the same encoder which was based on VGG16 to supply features to different decoders for classification, segmentation, and detection tasks. In segmentation decoder, the low-resolution segmentation feature map was convoluted and then upsampled using transposed convolution. It was claimed that MultiNet could perform inference at 23 fps. The structure of Up-Conv-Poly [12] was very similar to U-Net. It achieved MaxF score 93.83%. For images with size 500 × 500, this network could process each frame within 83 ms on TITAN X GPU.

Other CNN based road segmentation algorithms such as DEEP-DIG [13] and MAP [14] generated a precise drivable region but required heavy computational power.

In our previous work RoadNetV3 [15], we introduced Long-Short Term Memory (LSTM) to help finding the contour of the road. It extracted features via a FCN-like encoder. After that, several convolutional-LSTM layers followed to predict the contours of drivable region. It achieved 93.08% in accuracy but 300 ms per frame.

B. Real-Time Segmentation

In recent years, some researchers have shifted their focus to real-time segmentation tasks. Their solutions are generally categorized into two groups (Fig. 2), one is encoder-decoder network and the another one is bilateral network.

FPENet [16] adopted the encoder-decoder structure. By using a feature pyramid encoding block to encode multi-scale contextual features with depthwise dilated convolutions in all stages and a mutual embedding upsample module as decoder, FPENet efficiently aggregated of high-level semantic features and low-level spatial details. Through introducing an efficient spatial pyramid (ESP), ESPNet [17] brought great improvement in both speed and performance. In its improved version, ESPNet-V2 [18] further enlarged the receptive field and reduced the calculation of parameters. In [19], DABNet balanced the efficiency and accuracy via stacking lightweight blocks with different dilation rates. DFANet [20] aggregated multi-scale features from different layers to gain higher accuracy in spatial details. The lightweight backbone of DFANet guaranteed its real-time processing speed.

ContextNet [21] proposed the solution of bilateral structure for the first time. A deep but low-resolution network extracted the context information. And a shallow but high-resolution network focused on detailed spatial information. BiSeNet [22]

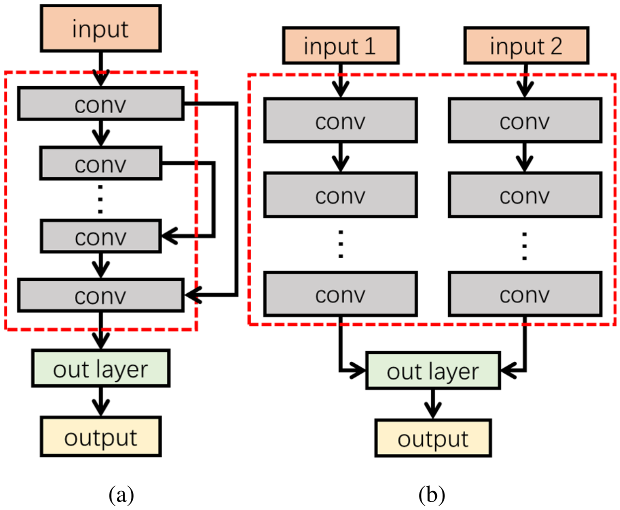


Fig. 2. The mainstream structures for real-time semantic segmentation. (a) illustrates the u-shape encoder-decoder structure and (b) demonstrates the bilateral structure.

inherited the solution of ContextNet and improved the feature fusion modules by creating attention residual module and feature fusion module. Via adding global pooling layer and residual layer, BiSeNet outperformed ContextNet. In ICNet [23], the authors borrowed the image pyramid thinking from PSPNet [24]. One more branch was added to acquire more spatial details. Plus, the label guided training for each branch, ICNet had better accuracy than BiSeNet but longer processing time. BiSeNet-V2 [25] improved the first version by replacing feature fusion module into aggregation module and using Seg Head to guide the loss of each feature extractor layer. Other networks like LBN-AA [26], CANet [27] also used similar structure.

Solutions other than the two mentioned above also represent good results. FarSee-Net [28] applied Cascaded Factorized Atrous Spatial Pyramid Pooling (CF-ASPP) at the end of feature extraction layers to guarantee enough spatial information was captured. What is more, to reduce the number of operations, sub-pixel convolution was deployed, so that FarSee-Net accepted low-resolution input and generated high-resolution output.

C. FPGA Implementation of Segmentation

To accelerate the inference speed, a great amount of effort focused on FPGA implementation of segmentation neural networks. The key to hardware accelerator for CNNs was the trade-off between bandwidth and computation capability. U-Net [7] and FCN [29] are both implemented in [30]. By utilizing convolution plus board removing method, this accelerator operated transposed convolution efficiently. Its performance was 107 GOPS and supported up to 17 fps for 512×512 images. A straight-forward fully convolution neural network for segmentation has been proposed and implemented on FPGA [31], [32]. Without changing the channel depth for each layer and skip connections used in U-Net [7], this accelerator pushed its performance to process 79.4 fps for input size $64 \times 180 \times 14$. Liu merged the convolution and transposed convolution into one vector multiplication unit and fused all intermediate feature maps in on-chip memory [33]. And the FPGA implementation reached 1578 GOPS, which was 57 fps for $256 \times 256 \times 3$ images. Another hardware architecture combining the convolution and transposed convolution operations was proposed in [34]. Its computation capability was 151.5 GOPS and 94.3 GOPS for convolution and transposed convolution respectively. besides, a 3D segmentation CNN accelerator was implemented in [35].

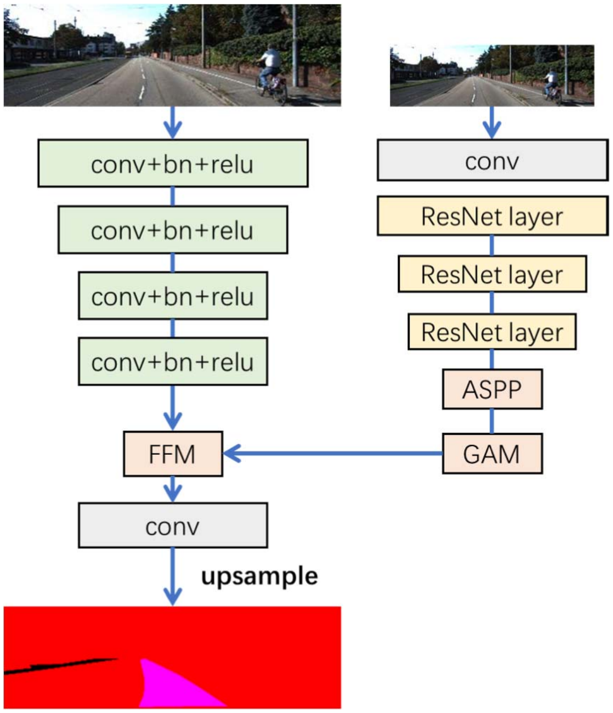


Fig. 3. Real-time road segmentation network RoadNet-RT structure.

III. Proposed Network

The proposed road segmentation network is inspired by ContextNet [21], BiSeNet [22] and ICNet [23]. It consists of two branches for context information and spatial information extraction respectively, as shown in Fig. 3.

The context branch is a deep network for extracting the context information, which consists of an input convolutional layer and two residual modules from ResNet18 [36]. Subsequently, the extracted features are fed to the ASPP module in order to concatenate the features from different fields of perception (dilated rates are 2, 4, 8 and 16, depth are 32 for each, in Fig. 4). Next, a Global Attention Module (GAM) is introduced to refine the context information. The GAM (Fig. 5a) is modified from the Attention Refinement Module in [22]. The GAM consists of a global average pooling layer together with a 1×1 convolutional layer who extracts global context feature. These refined global features are applied to context features via multiplication. The sigmoid layer decides whether to apply the global features or not. Since the context path does not have to focus on spatial details, we shrink the input image size by half in both width and height, as a step to further reduce the computation.

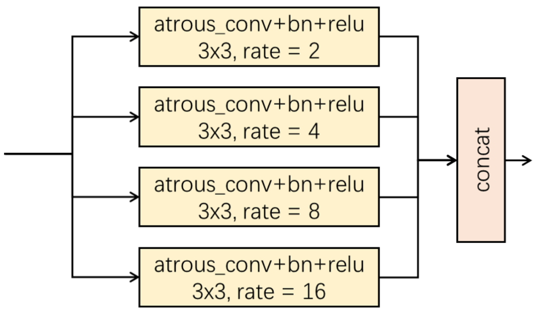
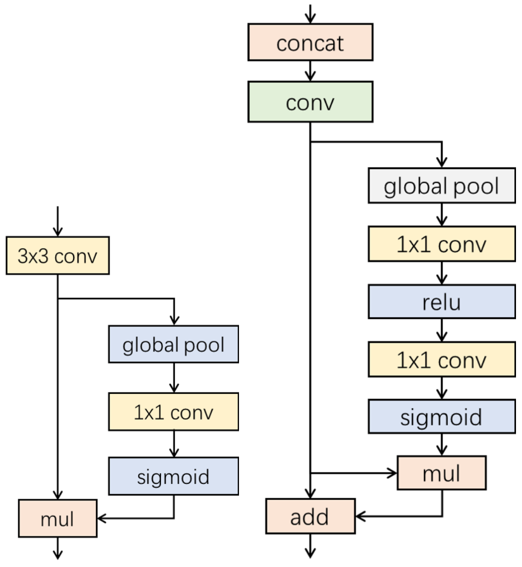


Fig. 4. Structure of ASPP.



(a) (b)

Fig. 5. (a) structure of GAM, (b) structure of FFM [22].

For spatial path, which focuses on spatial details of the input images, contains only four convolution layers. To enhance its capability of noticing details, no image resize is applied here. The context and spatial branches are fused in a residual refinement way, called Feature Fusion Module (FFM) [22] (Fig. 5b). The residual of FFM is the product of input feature map and its global attention path, including global average pooling layer, 1 × 1 convolutional layer, activation layers (ReLU and Sigmoid). At the end of the network, to reproduce the output with the same size as input, the output of FFM is upsampled 8 times by the bi-linear resize algorithm.

The number of channels is chosen to be factor of 32. This is based on the number of parallelisms the hardware accelerator could support, in order to maximize the efficiency of it.

A. Training Details

This road segmentation network is implemented using Keras and trained from scratch on a single GeForce GTX 1080 GPU. All the convolutional layers were initialized using the Xavier uniform initializer [37]. During training, the batch size is set to 24. The Adam optimizer works with learning rate 1e-3. When in plateau, a reduction rate of 0.8 is applied to the learning rate. A hybrid loss function combining Dice loss and Focal loss is deployed here expecting to balance the positive and negative samples.

TABLE I

KITTI Evaluation Comparison on URBAN_ROAD Benchmark.

Benchmark	MaxF	AP	FPR	FNR
RBANet [4]	96.30 %	89.72 %	2.75 %	2.50 %
SSLGAN [9]	95.53 %	90.35 %	2.28 %	4.76 %
RBNet [5]	94.97 %	91.49 %	2.79 %	4.99 %
StixelNet-II [10]	94.88 %	87.75 %	4.04 %	3.13 %
RoadNet-RT	92.55 %	93.21 %	3.86 %	7.84 %

TABLE II

Road Segmentation Results on the CamVid Test Dataset.

Methods	F1-measure	Precision	Recall
RBANet [4]	96.72%	97.14%	96.30%
RoadNet-RT	92.98%	94.70%	91.91%

Data augmentation for training includes random horizontal flip, Gaussian noise adding, random brightness contrast, random blurring, etc.

B. Dataset and Evaluation

1) KITTI: The dataset for training and evaluation is the KITTI road segmentation dataset, which contains 289 training images and 290 testing images. The training image size ranges from 370 × 1224 to 375 × 1242. The evaluation job is done by an online evaluation server supplied by KITTI. The evaluation (Tab.III is divided into Urban Unmarked (UU), Urban Marked (UM) and Urban Multiple Marked lanes (UMM). URBAN_ROAD is the comprehensive evaluation of the above three.

When running on GeForce GTX 1080 GPU, this network can process each image with 280 × 960 pixels in 9 ms. Four samples of predictions are demonstrated in front view and bird eye view by Fig. 6 and Fig. 7 respectively, where green area represents the overlap between prediction and ground truth, red area is road in ground truth but not correctly predicted by our network, and blue area is not road but recognized as road by our network.

Tab. I shows the performance comparison among RoadNet-RT and other state-of-the-art networks. The FNR (False Negative Rate) reflects the ratio of pixels, which are road but are wrongly recognized as non-road. While the FPR (False Positive Rate) calculates the ratio of pixels, which are non-road but are wrongly classified as road. From Tab.I, we can see RoadNet-RT has much higher FNR (7.84%) than the peers. Considering moving autonomous vehicles, high FNR would pose more restrictions on the drivable region. On the contrary, a high FPR means the neural network classifies more non-road pixels as road. For example, vehicles may recognize other cars on the roadside or bush as drivable region. Thus, high FPR would cause a safety issue. In FPR column of Tab.I, RoadNet-RT's FPR is comparable to the peers. Therefore, we consider Roadnet-RT is as safe as other state-of-the-art networks listed in Tab. I.

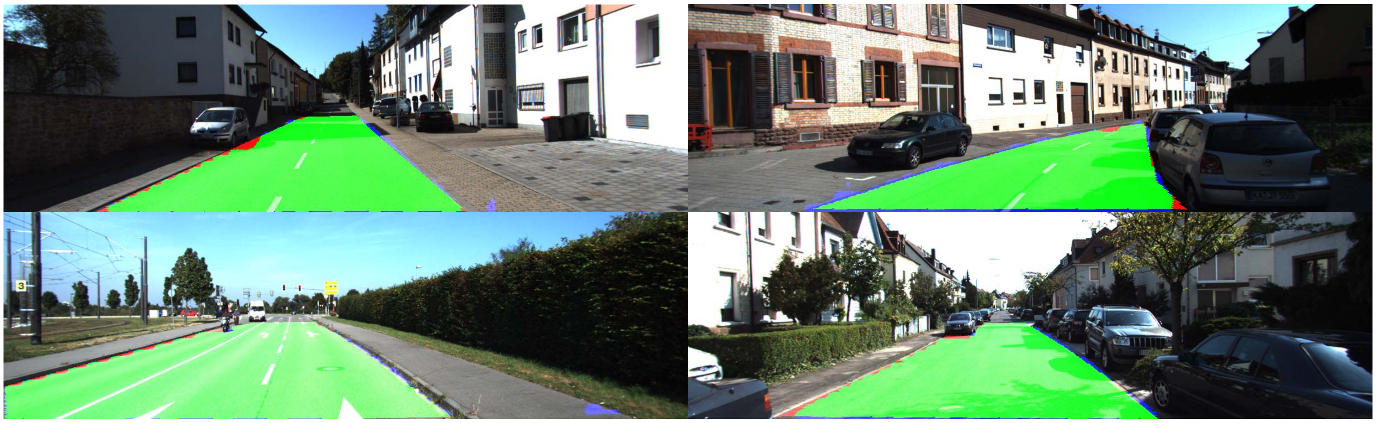


Fig. 6. Road segmentation results in camera view.

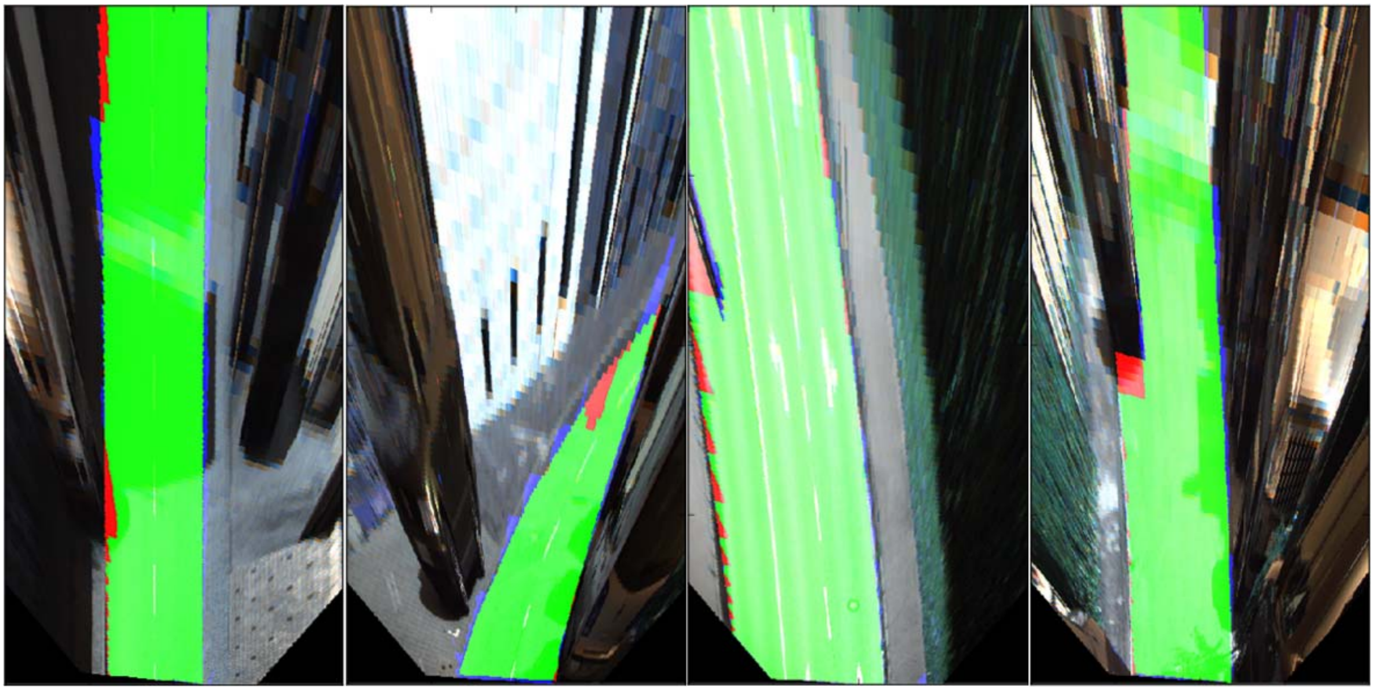


Fig. 7. Road segmentation results in Bird-Eye View.

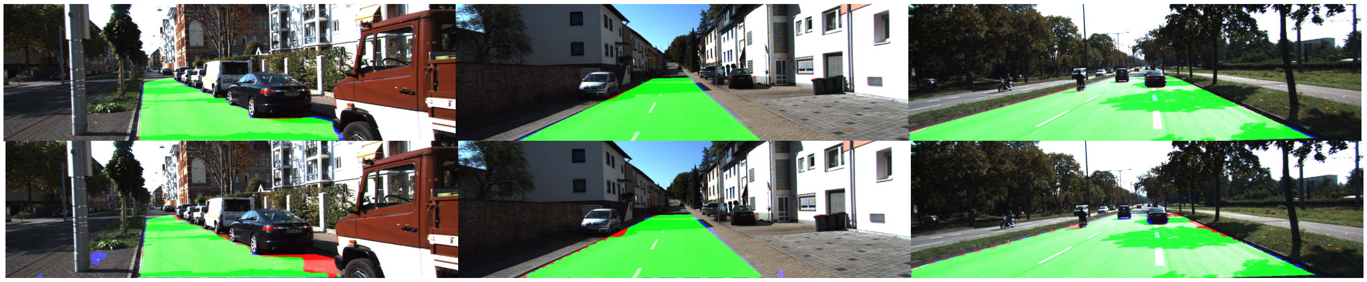


Fig. 8. Comparison between RBANet (top) and RoadNet-RT (bottom).

As shown in Fig. 8, comparing to RBANet, most of the classification errors of RoadNet-RT occurred near the boundary of the road since we choose not to include boundary attention in the model owing to the computation complexity. These errors won't affect autonomous driving due to path planning algorithm does not consider boundary of the drivable area.

2) CamVid: Besides the well-known KITTI dataset, the RoadNet-RT has also been evaluated on the CamVid dataset to verify its effectiveness on various road scenes (Tab. II). For

TABLE III
PERFORMANCE EVALUATION FROM KITTI ONLINE TEST SERVER.

Benchmark	MaxF	AP	PRE	REC	FPR	FNR
UM_ROAD	91.99%	92.54%	92.75%	91.24%	3.25%	8.76%
UMM_ROAD	93.98%	95.19%	94.47%	93.49%	6.01%	6.51%
UU_ROAD	90.79%	91.67%	91.79%	89.80%	2.62%	10.20%
URBAN_ROAD	92.55%	93.21%	92.94%	92.16%	3.86%	7.84%

F1 score, RoadNet-RT achieves 92.98% accuracy on CamVid test dataset, which is 3.74% less when comparing to the SOTA network RBANet [4]. Since the processing time of RBANet on CamVid is not provided in [4], we skip the processing speed comparison on CamVid.

IV. NETWORK OPTIMIZATION FOR HARDWARE

In this section, we summarize some guidelines to optimize specific CNNs toward FPGAs accelerator implementation. So that on-chip resources efficiency and computation efficiency FPGA design are maximized. Different from the conventional optimization techniques, the goal of this step is to balance the number of operations, number of weights and computation patterns, while remaining the accuracy within a reasonable range.

A. Depthwise Separable Convolution

Depthwise separable convolution is initially introduced in [38]. It has been widely adopted by a great number of lightweight neural networks such as Xception [39], MobileNet series [40], [41]. The main idea of depthwise separable convolution is to decompose standard convolution into a 3 × 3 depthwise convolution and a 1 × 1 pointwise convolution to achieve smaller number of weights and consequently less operations. Assuming D_K is the size of convolution kernel, M is the depth of input feature maps and N is the number of convolution kernels (also the channel number of output feature maps).

During depthwise convolution, a single filter is applied to each input channel. And then the pointwise convolution applies a 1 × 1 convolution to combine the outputs of the depthwise convolution. The number of weights required by standard convolution and depthwise separable convolution are calculated in (1) and (2) respectively.

$$D_K \cdot D_K \cdot M \cdot N \tag{1}$$

$$D_K \cdot D_K \cdot M + M \cdot N \tag{2}$$

Therefore, when replacing standard convolution with depthwise separable convolution, the reduction ratio of weights is

$$\frac{D_K \cdot D_K \cdot M + M \cdot N}{D_K \cdot D_K \cdot M \cdot N} = \frac{1}{N} + \frac{1}{D_K^2} \tag{3}$$

Besides the parameter reduction and operation number decreasing, from the hardware implementation point of view, depthwise separable convolution need not as large size accumulator as required by standard convolution. In standard convolution, every element of output feature map is the sum of $D_K \cdot D_K \cdot M$ elements. While in depthwise separable convolution, that is the sum of $D_K \cdot D_K$ and M elements for depthwise convolution and pointwise convolution respectively. On the other side, separating standard convolution into depthwise convolution and point convolution requires intermediate feature map buffering, and hence demands larger bandwidth.

TABLE IV
COMPARISON OF ROADNET-RT WITH AND WITHOUT DEPTHWISE SEPARABLE CONVOLUTION.

Convolution type	IOU[1]	parameters
Standard	93.67%	756,032
Depthwise separable	92.30%	133,870

Applying this to RoadNet-RT proposed in this article, the total number of parameters is reduced from 756K to 134K, which is illustrated in Tab. IV. Although the accuracy loss is 1.37%, the number of parameters reduces by a factor of 5.64.

B. Large Kernel Size Convolution

The most commonly used kernel size for convolution is 3 × 3. However, in order to have large size of field of perception, especially in the first layer, large kernel size is usually desired (7 × 7 in ResNet [36] for instance).

Algorithm 1 Cascaded Loop of Standard Convolution

```
for no in Nof do                          ▷ output channel,loop-4
    for (y,x) in (Noy,Nox) do             ▷ feature map,loop-3
        for ni in Nif do                  ▷ input channel,loop-2
            for (ky,kx) in (K,K) do       ▷ kernel,loop-1
                $F_{out}$[no,y,x]+=
                $F_{in}$[ni,y-ky,x-kx] *$K$[no,ni,ky,kx]
            $F_{out}$ += $bias$[no]
```

However, to deal with different kernel size filters affects either parallelism of processing or the efficiency of buffer usage. From matrix multiplication point of view (in Alg. 1), through keeping the loop-1, hardware accelerator can handle different size of filters without extra multipliers consumed. But the penalty is the parallelism of loop-1 loss. However, different size of filter requires different size of on-chip memory. Consider a feature map with size $W \cdot H \cdot C$, to buffer it for $K \cdot K$ filter, memory size $(W+K-1)\cdot(H+K-1)\cdot C$ is need. So that

[1] Since KITTI online test sever limits the submission to be 3 times per month, therefore 20% of the training set has been split as validation set to evaluate the methods we proposed. Here we choose IOU as the main metric to estimate the performance of different methods. IOU is one of the most important and the most widely used metrics for segmentation performance evaluation.

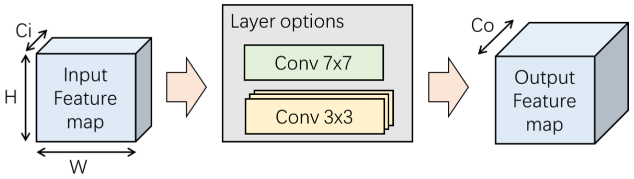


Fig. 9. Strategy for large convolutional layer replacement.

TABLE V

COMPARISON BETWEEN 7 × 7 CONVOLUTION AND ITS REPLACEMENT (C_i IS THE INPUT FEATURE MAP CHANNEL NUMBER AND C_o IS THE OUTPUT FEATURE MAP CHANNEL NUMBER, THEY EQUAL 32 AND 64 RESPECTIVELY IN THIS EXPERIMENT).

Method	IOU	parameter
1 conv 7 × 7	93.67%	$7\cdot7\cdot C_i\cdot C_o$
3 conv 3 × 3	93.67%	$3\cdot3\cdot C_i\cdot C_o$+$3\cdot3\cdot C_o\cdot C_o + 3\cdot3\cdot C_o\cdot C_o$

TABLE VI

PERFORMANCE COMPARISON BETWEEN DILATED CONVOLUTION (3 × 3 WITH DILATED RATE 3) AND ITS REPLACEMENT.

Method	IOU	parameter
1 conv 3 × 3 dilated rate 3	93.67%	$3\cdot3\cdot C_i\cdot C_o$
3 conv 3 × 3	93.66%	$3\cdot3\cdot C_i\cdot C_o$+$3\cdot3\cdot C_o\cdot C_o + 3\cdot3\cdot C_o\cdot C_o$

the feature map buffer for 7×7 filter is $4\cdot(W+H+4)/(W\cdot H)$ times larger than that for 3 × 3 filter.

To pursue the same perceptive field of 7×7, three cascaded convolutional layers with kernel size 3 × 3 can replace one convolutional layer with kernel size 7 × 7. If so, there is no extra resource needed including both multipliers and memory. Besides, the number of operations decreases. As illustrated in Fig. 9, for input feature map size $W \cdot H \cdot C_i$ and output feature map size $W \cdot H \cdot C_o$, if 7 × 7 filter is applied, totally $(W \cdot H \cdot 7 \cdot 7 \times C_i \cdot C_o) = 49 \cdot W \cdot H \cdot C_i \cdot C_o$ GOPS costs. In case of three 3×3 convolutional layers, $3\cdot(W \cdot H \cdot 3 \cdot 3 \cdot C_i \cdot C_o)) = 27 \cdot W \cdot H \cdot C_i \cdot C_o$.

The performance comparison between these two options mentioned above is shown in Tab. V. When replacing the first convolutional layer (7 × 7) with three 3 × 3 convolutional layers, the accuracy loss in IOU is 0.19%. Since there is only one layer of 7 × 7 convolution, the save in operations and parameters are negligible.

In the segmentation networks, dilated convolution [42] is the most widely used method to enlarge the perceptive field without introducing more weights. Unfortunately, during convolution with dilated kernel (3×3 with dilated rate equals 3 for instance), the region required from feature map is still 7 × 7. This will introduce the dilemma described above still. The only difference is, if using three 3×3 convolutional layers instead of one dilated 3 × 3 convolutional layers with dilated rate as 3, two times more weights and two times more operations are unavoidable. However, since the dilated convolutional layer usually won't dominant, this penalty is still affordable.

C. Consideration of Channel Depth

TABLE VII

THE PERFORMANCE COMPARISON WITH AND WITHOUT BN LAYER, BOTH OF THEM ARE TRAINED USING THE SAME BATCH SIZE AND THE SAM GPU.

Method	with BN	without BN
IOU	93.67%	93.39%
converge@epoch	350	340
duration/epoch	10s	8s

In our hardware implementation, after considering the given resources on ZCU102 board, loop-2 in Alg. 1 has been unrolled with 32 feature maps processed in parallel. To maximum the computation efficiency of accelerator, it's better that the input feature map depth of all layers align to integer factor of 32.

D. Batch Normalization

During inference, Batch Normalization (BN) is downgraded into 1 × 1 convolution and further merged into convolutional layer prior than it. The merged weights and bias follow (4) and (5), where W and b represent weights and bias respectively.

$$W_{merge} = W_{BN} \cdot W_{conv} \quad (4)$$

$$W_{merge} = W_{BN} \cdot b_{conv} + b_{BN} \quad (5)$$

Batch normalization layer is helpful for fast convergence but not always a necessary layer concerning to the accuracy (PointNet [43] for instance). The contribution of BN layer is evaluated in Tab.VII, from which we find in our segmentation neural network, BN helps to increase the accuracy by 0.28% without too much difference in convergence. Therefore, BN layers are kept in RoadNet-RT.

Some experiments declared that BN after ReLU usually shows better result [44]. But this may vary from one network to another.

E. Quantization

To maximize the computation capability of FPGA, fixed point operations is preferred. Quantization aware training has been performed for 8-bit and 16-bit respectively with the help of model optimization library from QKeras [45]. Brute-force quantization may lead to unacceptable precision loss. While quantization aware training restricts the bit-width during training. This not only compensates the precision loss but introduces more non-linearity.

The performance after quantization is shown in Tab. VIII. The IoU accuracy of 8-bit implementation is 92.36%, while that of 16-bit quantization is 92.40%. The accuracy of 16-bit quantization is 0.04% higher than that of 8-bit quantization, but it requires twice much memory for weights storage. Here we choose the 8-bit INT quantization for hardware implementation, 1) from storage perspective, memory space for 8-bit weights is only half of that for 16-bit quantization, 2) from hardware resources perspective, each DSP48E2 core could perform two 8-bit multiplications simultaneously but only one for 16-bit multiplication [46].

TABLE VIII

PERFORMANCE OF 8-BIT AND 16-BIT QUANTIZED NETWORKS.

Bit Width	IOU	size of parameters
float32	93.67%	2.88MB
int16	92.40%	1.44MB
int8	92.36%	0.72MB

TABLE IX

PERFORMANCE COMPARISON FOR DIFFERENT TECHNIQUES.

Technique	IOU
No optimization	93.67%
opt 1 - Replace 7×7 kernel	93.67%
opt 2 - Replace dilated convolution	93.62%
opt 3 - Depthwise separable convolution	93.07%
opt 4 - Quantization (INT8 on FPGA)	91.99%

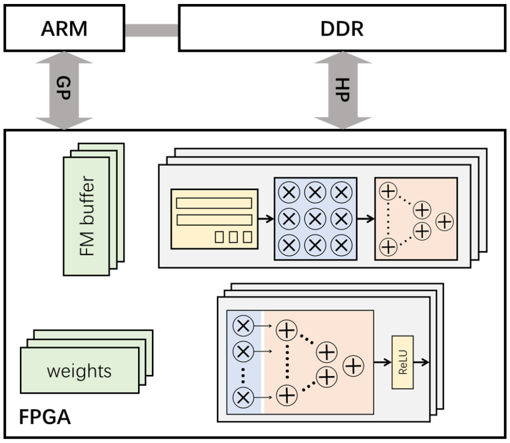


Fig. 10. System overview of RoadNet-RT accelerator.

F. Progressive Impact of Optimization Techniques

Considering the impact on precision loss, all the optimization (opt) techniques described above have been applied to RoadNet-RT progressively. The corresponding changes in IOU precision are listed in Tab. IX. As mentioned earlier, 7×7 kernel can be computed using 3×3 convolutions and there is no degradation of accuracy. Next, we use 3×3 convolution to replace dilated convolution with different dilated rates. There are four dilated convolutions in RoadNet-RT, which accounts for a performance drop to 0.05%. Depthwise separable convolution sharply compresses the computation complexity at the penalty of reduced network capacity, resulting an additional (0.55%) precision loss. Finally we apply fixed-point quantization to the model, which contributes to the largest precision loss (1.08%) among all optimization techniques.

V. SYSTEM-ON-CHIP IMPLEMENTATION

To fully utilize the computation resources, the whole system is partitioned into software part (done by ARM processor) and hardware part (running on FPGA). The software part job is image resize for both input and output of neural network (Fig. 3). With the help of OpenCV library [47], image resize can be easily done on PYNQ platform.

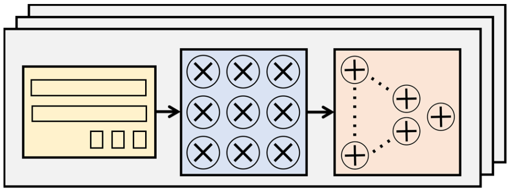


Fig. 11. Block diagram of depthwise convolution module.

Combining the depthwise and pointwise convolution into one process engine array is possible, but may result in output feature map reshape before sending to DDR memory and consequently decrease the efficiency of the accelerator. Thereby, we decide to separately implement these two computation modules. The overview of hardware architecture is demonstrated in Fig. 10. It consists of depthwise convolution module, and pointwise convolution module, feature map buffers, weights buffers. A finite state machine controls the running order of CNN operations. All the modules mentioned above are configurable based on the on-chip resources available on the target FPGA platform.

We chose 32 as the depth of process engine array, due to 1) target ZCU102 development kit supplies 2520 DSPs and 32.1Mb BRAM, which is sufficient for 32 process engines and corresponding feature map buffers 2) considering except the input layer and output layer, the depth of all the layers in RoadNet-RT are the product of 32, therefore using 32 can maximize the utilization of each multiplier, 3) as the greatest common divisor, using 32 as depth can minimize the data transporting for convolutions whose depth is large.

A. Depthwise Convolution Module

Depthwise convolution module (Fig. 11) contains line buffers, process engines (PEs) and adder trees. As descried in the previous section, to unroll the kernel loop (loop-1 in Alg. 1), line buffer is needed to generate the sliding patch. Since kernel size of all the convolutional layers in this segmentation network is 3×3, a multiplier array with length equal to 9 follows the line buffer. Correspondingly, an adder tree in the end sums the products up. To balance the computation efficiency and on-chip resources, the batch size of depthwise convolution module is set to 32.

B. Pointwise Convolution Module

To align to the depthwise convolution module to fit the same size of feature buffers, the pointwise convolution module (Fig. 12) is designed to handle 32×1 vector - 32×32 matrix multiplication. There are 3 components multiplier array, adder tree, and ReLU module form the Pointwise convolution module. If the batch normalization layer is placed before ReLU layer, it can be merged and completed by multiplier array and adder tree. Otherwise, 1 extra multiplier and 1 extra adder is necessary to perform the batch normalization operation.

C. GAM Module and FFM Module

Both GAM and FFM modules require operations with totally different computation patterns. Global average pooling

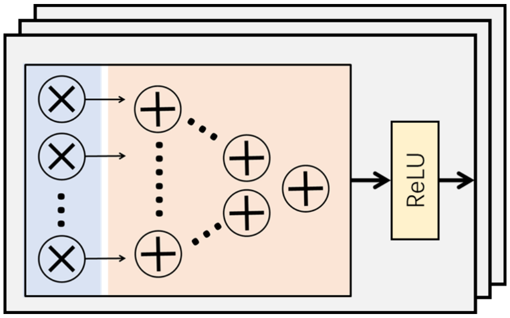


Fig. 12. Block diagram of pointwise convolution module.

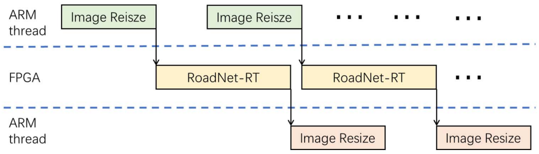


Fig. 13. Task partitioning of RoadNet-RT on SoC.

is to calculate the average value of one entire channel. Therefore, an accumulator plus one multiplier for each channel has been implemented. The following 1 × 1 convolution is mathematically vector-matrix multiplication, which can be either routed into pointwise convolutional module or implemented with extra resource, given the resource consumption of this operation is small. Sigmoid function is approximated by the piece-wise function and implemented using a Look-Up Table.

D. Buffers

The on-chip memory are divided into buffers for feature maps, weights and global pooling result respectively. In this design, 1) there is no biases, so that no extra buffer is needed for bias storage, and 2) since the weights occupy only small portion of the on-chip memory, so that they can be hard coded into on-chip memory.

To boost the processing speed, one effective way is to reduce the number of time data transmission (between FPGA and DDR memory). Multiple feature map buffers with size $35 \times 120 \times 32$ have been implemented as ping-pong buffers to decrease data swap as much as possible.

E. Tasks on ARM Processor

Referring to Fig. 10, the entire CNN is implemented on FPGA side. In order to fully utilize the available computation resources on SoC, the rest of the task has been assigned to the ARM processor. Thus, the whole RoadNet-RT are partitioned to both ARM processor and FPGA as shown in Fig. 13. All the three tasks are overlapped and pipelined, and this consequently speeds up the system speed.

VI. Results and Discussion

The implementation tools used in this article are Xilinx Vivado HLS and MATLAB HDL Coder Toolbox. The whole system has been implemented on ZCU102 development kit, with the PYNQ system installed (The system setup is show in Fig. 14). There are 548,160 Flip-Flops (FFs), 274,080 Look-Up Tables (LUTs), 1824 (32.1 Mb) Block RAMs (BRAMs) and 2,520 DSPs on the board. The FPGA resources consumption of this accelerator for both 16-bit and 8-bit quantization formats are shown in Tab. X.

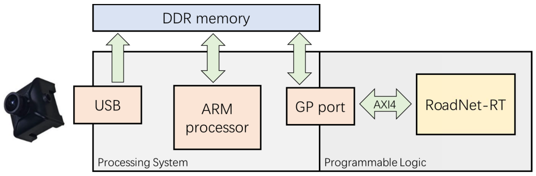


Fig. 14. Setup of the road segmentation system.

TABLE X
FPGA on-Chip Resource Usage of RoadNet-RT.

bitwidth	FF	LUT	DSP	BRAM
8-bit	115684	260335	1560	1340

TABLE XI
Performance Comparison Between Floating Point and Fixed Point.

Device	precision	accuracy (IOU)	processing time
GPU	float32	93.67%	9 ms
FPGA	int8	91.99%	5.24 ms

Since each DSP48E2 slice can handle two 8-bit×8-bit multiplication while the number for 16-bit number is one, thus 8-bit format accelerator consumes almost the same DSP slices and BRAMs as that in 16-bit format but twice the number of input images. To maximum the computation capability of hardware, we quantize all the weights into 8-bit. When running at 250 MHz, this 8-bit accelerator's processing speed is 196.7 fps. In Tab. XII, all the image-based road segmentation solutions in the KITTI leaderboard are summarized and compared to our solution in GPU and FPGA. Most of the existing methods cost 100 ms or longer. One of the only two real-time solutions FCN-LC [48] runs on TITAN X GPU, which requires 600-650W power supply on PC to support. Therefore, our solutions supply a well-balanced and practical way to run this the road segmentation task on embedded devices.

In this accelerator, there are 8 feature map buffers are allocated. But this number may vary according to the balance between available resources on the target FPGA and required processing speed. More feature map buffers can store more intermediate feature maps and consequently increase the processing speed. While less feature map buffers require more temporary data stored in external memory rather than on-chip ones. And thus leads to longer processing time.

The FPGA performance on the KITTI valid dataset is shown in Tab. XI. After replacing all the large kernel, dilated convolution into convolutions with uniform kernel size and quantization, when using INT8 format weights, the IOU of network on FPGA is 91.99%, which is 1.68% less than the proposed floating point RoadNet-RT.

VII. Conclusion

This article presents a real-time, high-throughput convolutional neural network architecture for road segmentation. Several optimization techniques are applied to reduce the number

TABLE XII

PERFORMANCE COMPARISON OF ALL THE IMAGE-BASED ROAD SEGMENTATION SOLUTIONS IN THE KITTI LEADERBOARD (BLANK MEANS IT IS NOT MENTIONED IN THE ORIGINAL PAPER).

Name	CNN-based	Input shape	Devices	Accuracy(MaxF)	Processing speed
RBANet[4]	✓	360 × 720	TITAN XP	96.30%	160 ms
SSLGAN[9]	✓	375 × 1242	TITAN X	95.53%	700 ms
RBNet[5]	✓	300 × 900	Tesla K20c	94.97%	180 ms
StixelNet-II[10]	✓	800 × 370	Quadro M6000	94.88%	1200 ms
MultiNet[11]	✓	1248 × 384		94.88%	170 ms
RoadNet3[15]	✓	600 × 160 × 5	GTX 950M	94.44%	300 ms
DEEP-DIG[13]	✓		Titan X	93.98%	140 ms
Up-Conv-Poly[12]	✓	500 × 500	TITAN X	93.83%	83 ms
OFA-Net[49]	✓			93.74%	40 ms
Up-Conv[12]	✓	300 × 300	GTX TITAN X	92.39%	52.2 ms
ALO-AVG-MM[50]	✓	624 × 192	GTX 1080	92.03%	29.6 ms
FTP[14]	✓			91.61%	280 ms
PT-ResNet[51]	✓		GTX 1080 Ti	91.61%	300 ms
FCN-LC[48]	✓	621 × 187	TITAN X	90.79%	30 ms
StixelNet[52]	✓	24 × 370		89.12%	1000 ms
MAP[14]	✓			87.80%	280 ms
SPRAY[53]	✓	800 × 600	GTX 580	87.09%	45 ms
multi-task CNN[54]	✓	375 × 1242	unknown type GPU	86.81%	25.1 ms
PGM-ARS[55]	✓	∼ 75 × 248	Intel i7-4700MQ processor	85.69%	50 ms
SRF[56]	✗	500 × 250		82.44%	200 ms
ARSL-AMI[57]	✗			80.36%	50 ms
CN[58]	✗			79.02%	2000 ms
Ours	✓	**280 × 960**	**GTX 1080**	**92.55%**	**9 ms**

of operations while preserving the accuracy performance. This networks achieves 92.55% MaxF score on KITTI dataset with 111 fps on GTX 1080 GPU (for image size 280×960). More importantly, using RoadNet-RT as an example, we present a systematic approach on how to perform CNN network optimization for hardware implementation. Following this as a guideline, one can easily convert any existing CNN structure into a computation efficient, high-throughput architecture for FPGA with little loss in accuracy. Several experiments have been conducted to support the proposed approach. In the end, a SoC design has been successfully demonstrated on ZCU102 FPGA development kit, which speeds up the processing time by a factor of 1.72 comparing to its GPU implementation.

REFERENCES

[1] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3D detection of vehicles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 3194–3200.

[2] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1951–1960.

[3] X. Cheng, P. Wang, C. Guan, and R. Yang, "CSPN++: Learning context and resource aware convolutional spatial propagation networks for depth completion," 2019, *arXiv:1911.05377*. [Online]. Available: http://arxiv.org/abs/1911.05377

[4] J.-Y. Sun, S.-W. Kim, S.-W. Lee, Y.-W. Kim, and S.-J. Ko, "Reverse and boundary attention network for road segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 876–885.

[5] Z. Chen and Z. Chen, "RBNeT: A deep neural network for unified road and road boundary detection," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2017, pp. 677–687.

[6] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3029–3037.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[9] X. Han, J. Lu, C. Zhao, S. You, and H. Li, "Semisupervised and weakly supervised road detection based on generative adversarial networks," *IEEE Signal Process. Lett.*, vol. 25, no. 4, pp. 551–555, Apr. 2018.

[10] N. Garnett *et al.*, "Real-time category-based and general obstacle detection for autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 198–205.

[11] M. Teichmann, M. Weber, M. Zollner, R. Cipolla, and R. Urtasun, "MultiNet: Real-time joint semantic reasoning for autonomous driving," in *Proc. 4th IEEE Intell. Vehicles Symp.*, Jun. 2018, pp. 1013–1020.

[12] G. L. Oliveira, W. Burgard, and T. Brox, "Efficient deep models for monocular road segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4885–4891.

[13] J. Munoz-Bulnes, C. Fernandez, I. Parra, D. Fernandez-Llorca, and M. A. Sotelo, "Deep fully convolutional networks with random data augmentation for enhanced generalization in road detection," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 366–371.

[14] A. Laddha, M. K. Kocamaz, L. E. Navarro-Serment, and M. Hebert, "Map-supervised road detection," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 118–123.

[15] Y. Lyu, L. Bai, and X. Huang, "Road segmentation using CNN and distributed LSTM," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[16] M. Liu and H. Yin, "Feature pyramid encoding network for real-time semantic segmentation," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–13.

[17] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNeT: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 552–568.

[18] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "ESPNetv2: A light-weight, power efficient, and general purpose convolutional neural network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9190–9200.

[19] G. Li and J. Kim, "DABNeT: Depth-wise asymmetric bottleneck for real-time semantic segmentation," in *Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–12.

[20] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep feature aggregation for real-time semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9522–9531.

[21] R. P. K. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring context and detail for semantic segmentation in real-time," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018, p. 146.

[22] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 325–341.

[23] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, “ICNeT for real-time semantic segmentation on high-resolution images,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 405–420.

[24] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.

[25] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, “BiSeNet v2: Bilateral network with guided aggregation for real-time semantic segmentation,” 2020, *arXiv:2004.02147*. [Online]. Available: http://arxiv.org/abs/2004.02147

[26] G. Dong, Y. Yan, C. Shen, and H. Wang, “Real-time high-performance semantic image segmentation of urban street scenes,” *IEEE Trans. Intell. Transp. Syst.*, early access, 2020, doi: 10.1109/TITS.2020.2980426.

[27] Q. Tang, F. Liu, J. Jiang, and Y. Zhang, “Attention-guided chained context aggregation for semantic segmentation,” 2020, *arXiv:2002.12041*. [Online]. Available: http://arxiv.org/abs/2002.12041

[28] Z. Zhang and K. Zhang, “FarSee-Net: Real-time semantic segmentation by efficient multi-scale context aggregation and feature space super-resolution,” 2020, *arXiv:2003.03913*. [Online]. Available: http://arxiv.org/abs/2003.03913

[29] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[30] S. Liu, H. Fan, X. Niu, H.-C. Ng, Y. Chu, and W. Luk, “Optimizing CNN-based segmentation with deeply customized convolutional and deconvolutional architectures on FPGA,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, pp. 1–22, Dec. 2018.

[31] Y. Lyu, L. Bai, and X. Huang, “Real-time road segmentation using LiDAR data processing on an FPGA,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[32] Y. Lyu, L. Bai, and X. Huang, “ChipNet: Real-time LiDAR processing for drivable region segmentation on an FPGA,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 5, pp. 1769–1779, May 2019.

[33] S. Liu and W. Luk, “Towards an efficient accelerator for DNN-based remote sensing image segmentation on FPGAs,” in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 187–193.

[34] L. Bai, Y. Lyu, and X. Huang, “A unified hardware architecture for convolutions and deconvolutions in CNN,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[35] J. Shen, D. Wang, Y. Huang, M. Wen, and C. Zhang, “Scale-out acceleration for 3D CNN-based lung nodule segmentation on a multi-FPGA system,” in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.

[36] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[37] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[38] L. Sifre and S. Mallat, “Rigid-motion scattering for image classification,” Ph.D. dissertation, Dept. d’Informatique, Ecole Normale Superieure, Paris, France, Oct. 2014.

[39] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[40] A. G. Howard *et al.*, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[42] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” 2015, *arXiv:1511.07122*. [Online]. Available: http://arxiv.org/abs/1511.07122

[43] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[44] (2016). *BatchNorm After ReLU*. Accessed: May 3, 2020. [Online]. Available: https://github.com/gcr/torch-residual-networks/issues/5

[45] *Qkeras: A Quantization Deep Learning Library for Keras*. Accessed: Dec. 6, 2019. [Online]. Available: https://github.com/google/qkeras

[46] Y. Fu, E. Wu, A. Sirasao, A. Attia, K. Khan, and R. Wittig, “Deep learning with int8 optimization on xilinx devices,” Xilinx, San Jose, CA, USA, White Paper WP486, 2017.

[47] G. Bradski, “The OpenCV library,” *Dr. Dobb’s J. Softw. Tools*, 2000. [Online]. Available: https://github.com/opencv/opencv/wiki/CiteOpenCV

[48] C. C. T. Mendes, V. Fremont, and D. F. Wolf, “Exploiting fully convolutional neural networks for fast road detection,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 3174–3179.

[49] S. Zhang, Z. Zhang, L. Sun, and W. Qin, “One for all: A mutual enhancement method for object detection and semantic segmentation,” *Appl. Sci.*, vol. 10, no. 1, p. 13, Dec. 2019.

[50] F. A. L. Reis, R. Almeida, E. Kijak, S. Malinowski, S. J. F. Guimaraes, and Z. K. G. do Patrocinio, “Combining convolutional side-outputs for road image segmentation,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[51] R. Fan *et al.*, “PT-ResNet: Perspective transformation-based residual network for semantic road image segmentation,” 2019, *arXiv:1910.13055*. [Online]. Available: http://arxiv.org/abs/1910.13055

[52] D. Levi, N. Garnett, and E. Fetaya, “StixelNet: A deep convolutional network for obstacle detection and road segmentation,” in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 1–109.

[53] T. Kuhnl, F. Kummert, and J. Fritsch, “Spatial ray features for real-time ego-lane extraction,” in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 288–293.

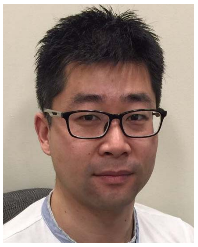
[54] M. Oeljeklaus, F. Hoffmann, and T. Bertram, “A fast multi-task CNN for spatial understanding of traffic scenes,” in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2825–2830.

[55] M. Passani, J. J. Yebes, and L. M. Bergasa, “Fast pixelwise road inference based on uniformly reweighted belief propagation,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2015, pp. 519–524.

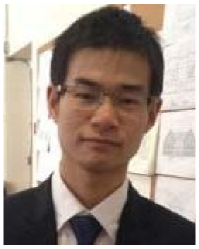
[56] L. Xiao, B. Dai, D. Liu, D. Zhao, and T. Wu, “Monocular road detection using structured random forest,” *Int. J. Adv. Robotic Syst.*, vol. 13, no. 3, p. 101, Jun. 2016.

[57] M. Passani, J. J. Yebes, and L. M. Bergasa, “CRF-based semantic labeling in miniaturized road scenes,” in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 1902–1903.

[58] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez, “Road scene segmentation from a single image,” in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2012, pp. 376–389.



Lin Bai (Graduate Student Member, IEEE) received the B.S. degree in integrated circuits design and integrated system from the University of Electronic Science and Technology of China in 2009, and the M.S. degree in electrical engineering and information technology from the Swiss Federal Institute of Technology, Zürich, in 2012. He is currently pursuing the Ph.D. degree with the Worcester Polytechnic Institute, USA. He was an FPGA engineer in industry. His current research interest includes the hardware acceleration of deep learning algorithms on FPGA and ASIC.



Yecheng Lyu (Graduate Student Member, IEEE) received the B.S. degree from Wuhan University, China in 2012, and the M.S. degree from the Worcester Polytechnic Institute, USA, in 2015, where he is currently pursuing the Ph.D. degree in autonomous vehicles. His current research interests include sensor fusion, autonomous vehicle perception, and deep learning.

Xinming Huang (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Virginia Tech in 2001. He was a Member of Technical Staffs with the Bell Labs, Lucent Technologies. Since 2006, he has been a Faculty with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute (WPI), where he is currently a Chair Professor. His current research interest includes circuits and systems, with emphasis on autonomous vehicles, deep learning, the IoT, and wireless communications.