ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-time

Rudra P K Poudel
rudra.poudel@crl.toshiba.co.uk
Ujwal Bonde
ujwal.bonde@crl.toshiba.co.uk
Stephan Liwicki
stephan.liwicki@crl.toshiba.co.uk
Christopher Zach
christopher.m.zach@gmail.com

Toshiba Research Europe
Cambridge, UK

Abstract

Modern deep learning architectures produce highly accurate results on many challenging semantic segmentation datasets. State-of-the-art methods are, however, not directly transferable to real-time applications or embedded devices, since naïve adaptation of such systems to reduce computational cost (speed, memory and energy) causes a significant drop in accuracy. We propose ContextNet, a new deep neural network architecture which builds on factorized convolution, network compression and pyramid representation to produce competitive semantic segmentation in real-time with low memory requirement. ContextNet combines a deep network branch at low resolution that captures global context information efficiently with a shallow branch that focuses on high-resolution segmentation details. We analyse our network in a thorough ablation study and present results on the Cityscapes dataset, achieving 66.1% accuracy at 18.3 frames per second at full (1024×2048) resolution (23.2 fps with pipelined computations for streamed data).

1 Introduction

Semantic segmentation provides detailed pixel-level classification of images, which is particularly suited for autonomous vehicles and driver assistance, as these applications often require accurate road boundaries and obstacle detection [3, 7, 16]. Modern systems produce highly accurate segmentation results, but often at the cost of reduced computational efficiency. In this paper, we propose ContextNet to address competitive semantic segmentation for autonomous driving tasks, requiring real-time processing and memory efficiency.

Deep neural networks (DNNs) are becoming the preferred approach for semantic image segmentation in recent years. High performance segmentation methods adopt state-of-the-art classification architecture using fully convolutional network (FCN) [23] or encoder-decoder techniques [2]. In particular, DeepLab [5] employs an increased number of layers to extract complex and abstract features, leading to increased accuracy. PSPNet [28] combines multiple levels of information through context aggregation from multiple feature resolutions, and

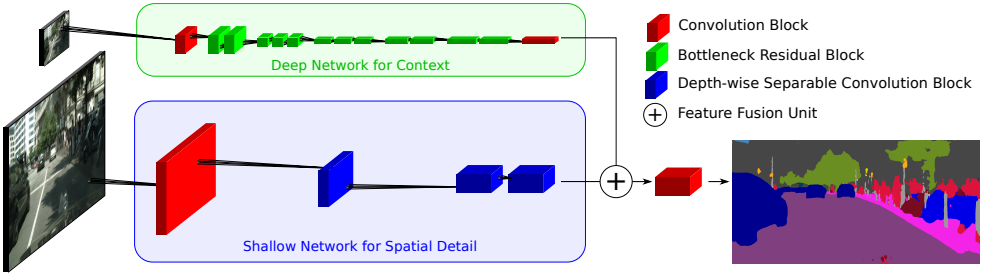

Figure 1: ContextNet combines a deep network at small resolution with a shallow network at full resolution to achieve accurate and real-time semantic segmentation.

benchmarks as one of the most accurate DNNs. The accuracy of these architectures comes at a high computational cost. Semantic segmentation of a single image requires more than a second, even on modern high-end GPUs (*e.g.* Nvidia Titan X) and hinders their deployment for driverless cars.

Recent interest in embedded devices, wearable devices and autonomous vehicles has sparked growing focus on semantic segmentation in real-time with low energy and memory requirements [2, 13, 17, 18, 19, 26, 27]. Common techniques include convolutional factorization [6, 12, 22], network pruning [10, 14] and weight quantization [13, 18, 26].

Convolution Factorization: Ordinary convolutions perform cross-channel and spatial correlations simultaneously. In contrast, convolution factorization employs multiple suboperations to reduce computation cost and memory. Examples include Inception [24], Xception [6] and MobileNet [12, 22]. In particular, MobileNet [12] decomposes a standard convolution into a depth-wise convolution (also known as spatial or channel-wise convolution) and a 1×1 point-wise convolution. MobileNetV2 [22] further improves this framework by introducing a bottleneck block and residual connections [11].

Network Compression: Network compression is orthogonal to convolution factorization. Network hashing or pruning is applied to reduce the size of a pre-trained network, resulting in faster test-time execution and a smaller parameter set and memory footprint [10, 14].

Network Quantization: The runtime of a network can be further reduced using quantization techniques [13, 18, 26]. These techniques reduce the size/memory requirements of a network by encoding parameters in low-bit representations. Moreover, runtime is further improved if binary weights and activation functions are employed since efficient XNOR and bit-count operations replace costly floating point operations of standard DNNs.

Despite these known techniques, notably, only ENet [17] implements a network that runs in real-time on the full resolution images of the popular Cityscapes dataset [7], but with significantly reduced accuracy in comparison to the state-of-the-art.

1.1 Contributions

In this paper we introduce ContextNet, a competitive network for semantic segmentation running in real-time with low memory footprint (Figure 1). ContextNet builds on the following two observations of previous work:

1. An increased number of layers helps to learn more complex and abstract features, leading to increased accuracy but also increased running time [5, 11, 28].

2. Aggregation of context information from multiple resolutions is beneficial, since it combines multiple levels of information for improved performance [23, 28].

Consequently, we propose a network architecture with branches at two resolutions. In order to obtain real-time performance, the network at low resolution is deep, while the network at high resolution is rather shallow. As such, our network perceives context at low resolution and refines it for high resolution results. The architecture is designed to ensure the low resolution branch is mainly responsible for providing the correct label, whereas the high-resolution branch refines the segmentation boundaries.

Our design is related to pyramid representations [4] which have been employed recently in many DNNs. RefineNet [15] uses multiple paths over which information from different resolutions is carefully combined to obtain high-resolution segmentation results. Ghiasi and Fowlkes [9] use a class specific multi-level Laplacian pyramid reconstruction technique for the segmentation task. However, neither achieve real-time performance.

ICNet [27] employs a subset of ResNet [11] at three resolution levels (full, half and one fourth of the original input resolution) which are later combined to provide semantic segmentation results. Here we emphasize, while our implementation of ICNet confirms accuracy in [27], the reported runtime is achieved only at half resolution images of Cityscapes [7]. In contrast, we experimentally show that it is possible to capture global context (semantically rich features) with only a single deeper network on smaller input size and local context (detailed spatial features) with a shallow network on full resolution.

We employ efficient *bottleneck residual blocks* [22] and *network pruning* [10, 14] to present a novel DNN architecture for real-time semantic segmentation with full floating point operations. Network quantization is not explored and is left for future work. In the following sections, we provide design choices of our proposed ContextNet and describe our detailed ablation analysis and experiments on Cityscapes [7].

2 Proposed Context Network (ContextNet)

The proposed ContextNet is visualized in Figure 1. ContextNet produces cost efficient accurate segmentation at low resolution, which is then combined with a sub-network at high resolution to provide detailed segmentation results.

2.1 Motivation

Combining different levels of context information is advantageous for the semantic segmentation task [5, 20, 23]. PSPNet [28] employs an explicit pyramid pooling module to improve performance by capturing global and local context at different feature resolutions.

Another noticeable trend is that state-of-the-art DNNs have grown deeper (*e.g.* [5, 11, 28]), since this can capture more complex and abstract features, and increase the receptive field. Unfortunately, higher number of layers ultimately increase runtime and memory requirements.

ContextNet combines both, deep networks and multi-resolution architectures. In order to achieve fast runtime we restrict our multi-scale input to two branches, where global information at low resolution is refined by a shallow network at high resolution to produce the final segmentation results in real-time.

Input	Operator	Output
$h \times w \times c$	Conv2d $1/1, f$	$h \times w \times tc$
$h \times w \times tc$	DWConv $3/s, f$	$\frac{h}{s} \times \frac{w}{s} \times tc$
$\frac{h}{s} \times \frac{w}{s} \times tc$	Conv2d $1/1, -$	$\frac{h}{s} \times \frac{w}{s} \times c'$

Table 1: *Bottleneck residual block* transferring the input from c to c' channels with h height, w width, expansion factor t, convolution type kernel-size/stride s and non-linear function f.

2.2 Network Design

We now describe the main building blocks, the overall architecture and discuss our design.

2.2.1 Depth-wise Convolution to Improve Run-time

Depth-wise separable convolutions factorize standard convolution (Conv2d) into a depth-wise convolution (DWConv), also known as spatial or channel-wise convolution, followed by a 1×1 point-wise convolution layer [12]. Cross-channel and spatial correlation is therefore computed independently, which drastically reduces the number of parameters, resulting in fewer floating point operations and fast execution time.

ContextNet utilizes DWConv, as we design its two main building blocks accordingly (Figure 1). The sub-network with down-sampled input uses bottleneck residual blocks with DWConv [22] (Table 1). In the sub-network at full resolution depth-wise separable convolutions are directly employed. We omit the nonlinearity between depth-wise and point-wise convolutions in our full resolution branch, since it had limited impact on accuracy in our initial experiments.

2.2.2 Capturing Global and Local Context

ContextNet has two branches, one for full resolution ($h \times w$) and one for lower resolution (*e.g.* $\frac{h}{4} \times \frac{w}{4}$), with input image height h and width w (Figure 1). Each branch has different responsibilities; the latter captures the global context of the image, and the former provides the detail information for the higher resolution segmentation. In particular, our design choices are motivated as follows:

1. For fast feature extraction, semantically rich features are extracted only from the lowest possible resolution.

2. Features for local context are extracted separately from full resolution input by a very shallow branch, and are then combined with low-resolution results.

Hence, significantly faster computation of image segmentation is possible in ContextNet.

Capturing context: The detail structure of the lower resolution branch is shown in Table 2. This sub-network consists of two convolution layers and 12 bottleneck residual blocks. Similar to MobileNetV2 [22], we employ residual connections for bottleneck residual blocks when input and output are of the same size. While the low resolution branches of ICNet [27] requires 50 costly layers of ResNet [11], in ContextNet a total of only 38 highly efficient layers are used to describe global context.

Spatial detail: The sub-network of the full resolution branch is kept as shallow as possible and only consists of four layers. Its objective is to refine the results with local context. In particular, the number of feature maps are 32, 64, 128 and 128 respectively. The first layer

Input	Operator	Expansion Factor	Output Channels	Repeats	Stride
$256 \times 512 \times 3$	Conv2d	-	32	1	2
$128 \times 256 \times 32$	bottleneck	1	32	1	1
$128 \times 256 \times 32$	bottleneck	6	32	1	1
$128 \times 256 \times 32$	bottleneck	6	48	3	2
$64 \times 128 \times 48$	bottleneck	6	64	3	2
$32 \times 64 \times 64$	bottleneck	6	96	2	1
$32 \times 64 \times 96$	bottleneck	6	128	2	1
$32 \times 64 \times 128$	Conv2d	-	128	1	1

Table 2: Branch-4 for compressed input. Repeated block use stride 1 after first block/layer.

Branch-1	Branch-4
-	Upsample $\times$ 4
-	DWConv (dilation 4) $3/1, f$
Conv2d $1/1, -$	Conv2d $1/1, -$
add, f	

Table 3: Features fusion unit of ContextNet.

uses standard convolution while all other layers use depth-wise separable convolutions with kernel size 3×3. The stride is 2 for all but the last layer, where it is 1.

We use fusion unit shown in Table 3 to merge the features from both branches. Since runtime is of concern, we use feature addition instead of concatenation. Finally, we use a simple 1×1 convolution layer for the final soft-max based classification results.

2.2.3 Design Choices

We have conducted several initial experiments before deciding on the final model of ContextNet using train and validation sets of the Cityscapes dataset. We have found that the use of a pyramid pooling module [28] after the low-resolution branch increases accuracy. Also, learning global context using down-sampled input is more efficient than learning with asymmetric convolution (for examples, $k \times 1$, $1 \times k$, where $k = 5/7/9$) on higher resolution inputs. Class weight balancing technique did not help, when we increased batch-size to 16 or more.

Empirically, we found a weighted auxiliary loss for the low-resolution branch is beneficial. We think, the auxiliary loss ensures that meaningful features for semantic segmentation are extracted by the branch for global context, and are learned independently from the other branch. The weight of the auxiliary loss was set to 0.4. Following [5, 28], a cross-entropy loss is employed as auxiliary and final loss of ContextNet.

3 Experiments

In our evaluation, we present a detailed ablation study of ContextNet on the validation set of the Cityscapes dataset [7] , and report its performance on the Cityscapes benchmark.

3.1 Implementation Details

All our experiments are performed on a workstation with Nvidia Titan X (Maxwell, 3072 CUDA cores), CUDA 8.0 and cuDNN V6. We use ReLU6 as nonlinearity function due

	cn18	cn14	cn12	cn124	cn14-500	cn14-160
Accuracy (mIoU in %)	60.1	65.9	68.7	67.3	62.1	57.7
Frames per Second	21.0	18.3	11.4	7.6	20.6	22.0
Number of Parameters (in millions)	0.85	0.85	0.85	0.93	0.49	0.16

Table 4: ContextNet (cn14) compared to its version with half resolution (cn12) and eighth resolution (cn18) at the low resolution branch, and with multiple levels at quarter, half and full resolution (cn124) on Cityscapes validation set. Implementations with smaller memory footprint are also shown (cn14-500 and cn14-160).

to its robustness when used with low-precision computations [12]. During training, batch normalization is used at all layers and dropout is used before the soft-max layer only. During inference, parameters of batch normalization are merged with the weights and bias of parent layers. In the depth-wise convolution layers, we found that ℓ_2 regularization is not necessary, which is consistent with the findings in [12]. Since labelled training data is limited, we apply standard data augmentation techniques in all experiments: random scale 0.5 to 2, horizontal flip, varied hue, saturation, brightness and contrast.

The models of ContextNet are trained with TensorFlow [1] using RMSprop [25] with a discounting factor of 0.9, momentum of 0.9 and epsilon parameter equal to 1. Additionally, we apply a *poly learning rate* [5] with base rate 0.045 and power 0.98. The maximum number of epochs is set to 1000, as no pre-training is used.

Results are reported as mean intersection-over-union (mIoU) [7] and runtime considers single threaded CPU with sequential CPU to GPU memory transfer, kernel execution, and GPU to CPU memory exchange.

3.2 Cityscapes Dataset

Cityscapes is a large-scale dataset for semantic segmentation that contains a diverse set of images in street scenes from 50 different cities in Germany [7]. In total, it consists of 25,000 annotated $1024 \times 2048px$ images of which 5,000 have labels at high pixel accuracy and 20,000 are weakly annotated. In our experiments we only use the 5,000 images with high label quality: a training set of 2,975 images, validation set of 500 images and 1,525 test images which can be evaluated on the Cityscapes evaluation server. No pre-training is employed.

3.2.1 Ablation Study

In our ablation study weights are learned solely from the Cityscapes training set, and we report the performance on the validation set. In particular, we present effects on different resolution factors of the low resolution branch, introducing multiple branches, and modifications on the number of parameters. Finally, we analyse the two branches in detail.

Input resolution: The input image resolution is the most critical factor for the computation time. Our low resolution branch takes images of a quarter size at $256 \times 512px$ for Cityscapes images (denoted cn14). Alternatively, half (cn12) or one eighth (cn18) resolution could be used. Table 4 shows how the different options affect the results. Overall, larger resolution in the deeper context branch produce better segmentation results. However, these improvements come at the cost of running time.

Table 5 lists the IoU in more detail. As expected, accuracy of small-size classes (*i.e.* fence, pole, traffic light and traffic sign), classes with fine detail (*i.e.* person, rider, motorcycle

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
cn18	96.2	72.4	87.4	45.9	44.1	32.1	38.4	54.3	87.9	54.1	91.0	62.3	36.1	88.3	57.6	59.1	34.2	41.2	58.4
cn14	96.8	76.6	88.6	46.4	49.7	38.0	45.3	60.5	89.0	59.3	91.4	67.5	41.7	90.0	63.5	71.7	57.1	41.5	64.6
cn12	97.2	78.9	89.2	47.2	54.4	39.5	55.3	63.8	89.4	59.8	91.5	70.2	47.5	91.1	70.2	76.2	63.7	51.36	67.8
cn124	97.4	79.6	89.5	44.1	49.8	45.5	50.6	64.6	90.2	59.4	93.4	70.9	43.1	91.8	65.2	71.9	64.5	41.95	66.1

Table 5: Detailed IoU of ContextNet (cn14) compared to version with half (cn12) and eighth (cn18) resolution, and its multi-level version (cn124). Small-size classes (green), classes with fine detail (blue), and classes with very few samples (red) benefit from high resolution.

and bicycle) and rare classes (*i.e.* bus, train and truck) benefit from increased resolution at the global context branch. Other classes are often of larger size, and can therefore be captured at lower resolution. We conclude that cn14 is fairly balanced at 18.3 frames per seconds (fps) and 65.9% mIoU.

Multiple Branches: We designed ContextNet under consideration of runtime using only two branches. However, branches at multiple resolutions could be employed. In addition to previous results, Table 4 and Table 5 also include a version of ContextNet with two shallow branches at half and full resolution (cn124). In comparison to cn14, cn124 improves accuracy by 1.4% which confirms earlier results on multi-scale feature fusion [9, 15, 27]. Runtime however is reduced more than twice from 18.3 fps to 7.6 fps. Furthermore we note, cn12 which has a deep sub-network at half resolution outperforms cn124 in terms of accuracy and speed. These results show that using deep sub-network on higher resolution is more beneficial than on lower resolution. Further, the number of layers have positive effect on accuracy and negative effect on run-time. We therefore conclude that a two branch architecture is most promising in our implementation of ContextNet to run in real-time.

Number of Parameters: Apart from runtime, memory footprint is an important consideration for implementations on embedded devices. Table 4 includes two versions of ContextNet with drastically reduced number of parameters, denoted as cn14-500 and cn14-160. Surprisingly, ContextNet with only 159,387 and 490,459 parameters achieved 57.7% and 62.1% mIoU, respectively, on the Cityscapes validation set. These results show that the ContextNet design is flexible enough to adapt the computation time and memory requirements of the given system.

ContextNet vs Ensemble Nets: Global context with one fourth resolution and detail branches trained independently obtained 63.3% and 25.1% mIoU respectively. As expected, we found that context branch is not performing well on small-size classes and receptive field of the detail branch is not large enough for reasonable performance. Outputs (softmax) of context and detail networks are averaged to create an ensemble of networks, which are trained independently. Ensemble of both branches obtained 60.3% mIoU, which is 6.6% less than cn14 and 3% less than using the context branch alone. This provides further evidence that ContextNet architecture is a better choice for multi-scale features fusion.

Context Branch Analysis: We have zeroed the output of either the detail branch or the context branch to observe their individual contributions. The results are shown in Figure 2. In the first column, we see that the global context branch detects larger objects correctly (for example sky or trees) but fails around boundaries and thin regions. In contrast, the detail branch detects object boundaries correctly but fails at the centre region of objects. One exception is the centre region of trees, which is classified correctly, probably due to the

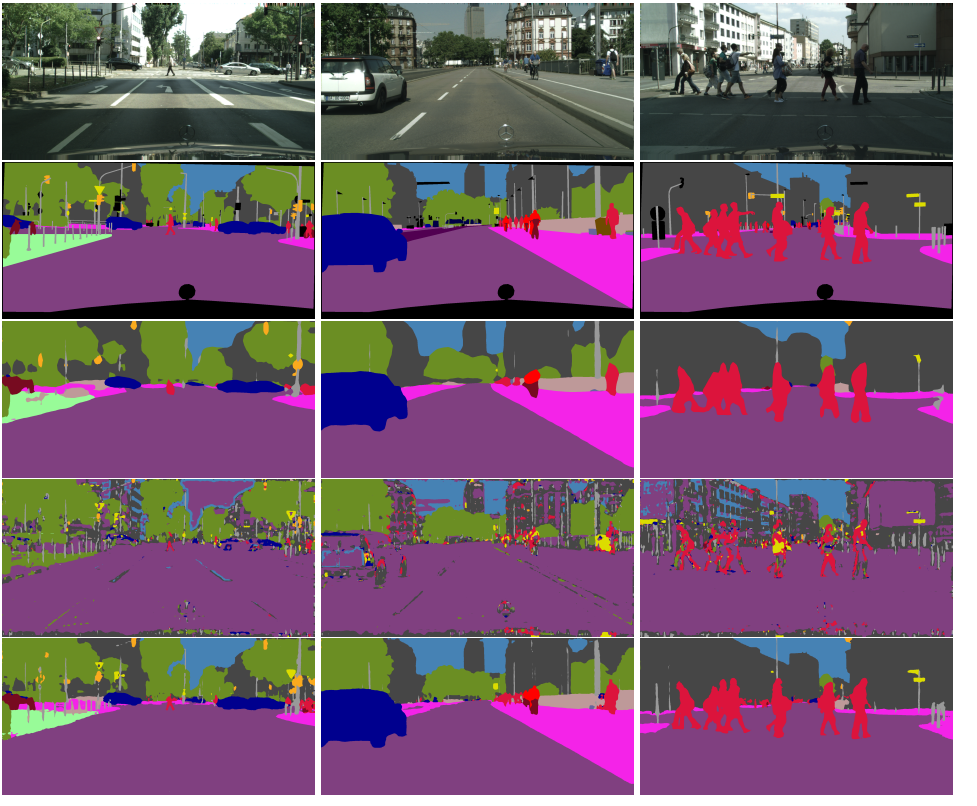

Figure 2: Visual comparison on Cityscape validation set [7]. First row: input RGB images; second row: ground truths; third row: context branch outputs; fourth row: detail branch outputs; and last row: ContextNet outputs using both context and detail branches. ContextNet obtained 65.9% mIoU, while global context with one fourth resolution and detail branches trained independently obtained 63.3% and 25.1% mIoU respectively

discriminative nature of tree texture. Finally, we can see that ContextNet correctly combines both information to produce improved results in last row. Similarly, in the middle column we can see that segmentation of the pedestrians are refined by ContextNet over the global context branch output. In the last column, even though detail branch detects poles and traffic signs, ContextNet fails to effectively combine some of these with the global context branch. Overall we observe that the context branch and the detail branch learn complementary information.

3.2.2 Cityscape Benchmark Results

We evaluate ContextNet on the withheld test-set of Cityscapes [7]. Table 6 shows the results in comparison to current state-of-the-art real-time segmentation networks (SegNet [2], ENet [17], ICNet [27] and ERFNet [19]), and offline methods (PSPNet [28] and DeepLab-v2 [5]). Table 7 compares the runtime at full, half and quarter resolution on images of Cityscapes. ContextNet achieves 64.2% before, and 66.1% mIoU after pruning (explained below), and runs at 18.3 fps in a single CPU thread of TensorFlow [1]. ENet [17] has a similar run-time

	Class mIoU (in %)	Category mIoU (in %)	Parameters (in millions)
DeepLab-v2 [5]*	70.4	86.4	44.__
PSPNet [28]*	**78.4**	**90.6**	65.7_
SegNet [3]	56.1	79.8	29.46
ENet [17]	58.3	80.4	00.37
ICNet [27]*	69.5	-	06.68
ERFNet [19]	68.0	86.5	02.1_
ContextNet (Ours)	66.1	82.7	00.85

Table 6: Cityscape benchmark results for the proposed ContextNet and similar networks. DeepLab-v2 [5] and PSPNet [28] are considered offline approaches. Runtime of other methods is shown in Table 7. (Methods with '*' are pre-trained on ImageNet [21].)

	1024×2048	512×1024	256×512
SegNet [3]*	1.6	-	-
ENet [17]*	20.4	76.9	142.9
ICNet [27]	14.2	46.3	83.2
ERFNet [19]*	11.2	41.7	125.0
ContextNet (Ours)	18.3	65.5	139.2
ContextNet (Ours)†	23.2	84.9	150.1

Table 7: Runtime on Nvidia Titan X (Maxwell, 3,072 CUDA cores) with TensorFlow [1], including sequential CPU/GPU memory transfer and kernel execution. (Results with '*' are taken from existing literature – it is not known if memory transfer is considered. Our measure with '†' denotes kernel execution time alone.)

but achieves only 58.3% accuracy. ICNet [27] and ERFNet [19] achieve 69.5% and 68.0% mIoU, respectively, but are considerably slower than ContextNet.[1]

We emphasize that our runtime evaluation includes the complete CPU and GPU pipeline including memory transfers. If parallel memory transfer and kernel execution are employed, our run time improves to 23.2 fps. Finally, in Table 7 we observe that ContextNet scales well for smaller input resolution sizes, and therefore can be tuned for the task at hand and the available resources. The results of ContextNet are displayed in Figure 2 for qualitative analysis. ContextNet is able to segment even small objects at far distances adequately.

Network Pruning: Network pruning is usually employed to reduce network parameters [14, 27]. Following the "lottery ticket" intuition [8] we start with wider networks (larger number of feature maps) and use pruning to obtain "skinnier" networks in order to increase the accuracy of our model. First, we train our network with twice the (target) number of feature maps to obtain improved results. We then decrease parameters progressively by pruning to 1.5, 1.25 and 1 times the original size. Following [14], we pruned filters with lowest ℓ_1 sum. Via pruning we improve the mIoU from 64.2% to 66.1% on the Cityscape test set.

4 Conclusions and Future Work

In this work we proposed a real-time semantic segmentation model called ContextNet, which combines a deep network branch at low resolution but with large receptive field with a shal-

[1] Although our implementation of ICNet [27] achieves similar accuracy, we do not achieve the timing mentioned by the authors. This might be caused by differences in software environment and the employed testing protocols.

low and therefore efficient full-resolution branch to enhance the segmentation details. ContextNet further extensively leverages depth-wise convolutions and bottleneck residual blocks for maximum memory and run-time efficiency.

Our ablation study shows that ContextNet effectively combines global and local context to achieve competitive result and outperforms other state-of-the-art real-time methods. We also empirically show that model pruning (in order to reach given targets of network size and real-time performance) leads to improved segmentation accuracy. Demonstrating that the ContextNet architecture is beneficial for other tasks relevant for autonomous systems such as single-image depth prediction is part of future work.

References

[1] M. Abadi and et. al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *TPAMI*, 2017.

[3] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recogn. Lett.*, 30(2), 2009.

[4] P. J. Burt and E. H. Adelson. The Laplacian Pyramid as a Compact Image Code. In *Readings in Computer Vision*, pages 671–679. 1987.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *arXiv:1606.00915 [cs]*, 2016.

[6] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv:1610.02357 [cs]*, 2016.

[7] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[9] G. Ghiasi and C. C. Fowlkes. Laplacian Pyramid Reconstruction and Refinement for Semantic Segmentation. *arXiv:1605.02264 [cs]*, May 2016.

[10] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *ICLR*, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, 2015.

[12] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]*, 2017.

[13] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized Neural Networks. In *NIPS*. 2016.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning Filters for Efficient ConvNets. In *ICLR*, 2017.

[15] G. Lin, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. In *CVPR*, 2017.

[16] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv:1606.02147 [cs]*, 2016.

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 2016.

[19] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *arXiv:1801.04381 [cs]*, 2018.

[23] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *PAMI*, 2016.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016.

[25] T. Tieleman and G. Hinton. Lecture 6.5-RMSProp, Coursera: neural networks for machine learning. *University of Toronto, Technical Report*, 2012.

[26] S. Wu, G. Li, F. Chen, and L. Shi. Training and Inference with Integers in Deep Neural Networks. In *ICLR*, 2018.

[27] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. *arXiv:1704.08545 [cs]*, 2017.

[28] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *CVPR*, 2017.