RGPNet: A Real-Time General Purpose Semantic Segmentation

A Preprint

Elahe Arani*, Shabbir Marzban*, Andrei Pata and Bahram Zonooz

Advanced Research Lab, NavInfo Europe, Eindhoven, The Netherlands
{elahe.arani, shabbir.marzban, andrei.pata, b.yoosefizonooz}@navinfo.eu

December 4, 2019

Abstract

We propose a novel real-time general purpose semantic segmentation architecture, called RGPNet, which achieves significant performance gain in complex environments. RGPNet consists of a light-weight asymmetric encoder-decoder and an adaptor. The adaptor helps preserve and refine the abstract concepts from multiple levels of distributed representations between encoder and decoder. It also facilitates the gradient flow from deeper layers to shallower layers. Our extensive experiments highlight the superior performance of RGPNet compared to the state-of-the-art semantic segmentation networks. Moreover, towards green AI, we show that using a modified label-relaxation technique with progressive resizing can reduce the training time by up to 60% while preserving the performance. Furthermore, we optimize RGPNet for resource-constrained and embedded devices which increases the inference speed by 400% with a negligible loss in performance. We conclude that RGPNet obtains a better speed-accuracy trade-off across multiple datasets.

1 Introduction

Convolutional neural networks (CNNs) have brought about a paradigm shift in the field of computer vision, leading to tremendous advances in many tasks [12, 13, 16, 17, 19, 31, 36]. Semantic segmentation, which associates each pixel to the object class it belongs to, is a computationally expensive task in computer vision [20]. Fast semantic segmentation is broadly applied to several real-time applications including autonomous driving, medical imaging and robotics [21, 23, 30, 33]. Accurate CNN-based semantic segmentation requires larger neural networks with deeper and wider layers. These larger networks are therefore not suitable for edge computing devices as they are cumbersome and require substantial resources.

Down-sampling operations, such as pooling and convolutions with stride greater than one, can help decrease the latency of deeper neural networks, however they result in decreased pixel-level accuracy due to the lower resolutions at deeper levels. Many recent approaches employ either encoder-decoder structure [1, 29, 35], a two or multi-branch architecture [26, 40, 43] or dilated convolutions [4–6, 44] to recover spatial information. While these real-time architectures perform appropriately on simple datasets, their performance is sub-optimal for complex datasets possessing more variability in terms of classes, sizes, and shapes. Thus, there is a significant interest in designing CNN architectures that can perform well on complex datasets and, at the same time, are mobile enough to be of practical use in real-time applications.

In this paper, we propose a real-time general purpose semantic segmentation network, RGPNet, that performs well on complex scenarios. RGPNet is based on an asymmetric encoder-decoder structure with a new module called *adaptor* in the middle. The adaptor utilizes features at different abstraction levels from both the encoder and decoder to improve the feature refinement at a given level allowing the network to preserve deeper level features with higher spatial resolution. Furthermore, the adaptor enables a better gradient flow from deeper layers to shallower layers by adding shorter paths

*Equal contribution

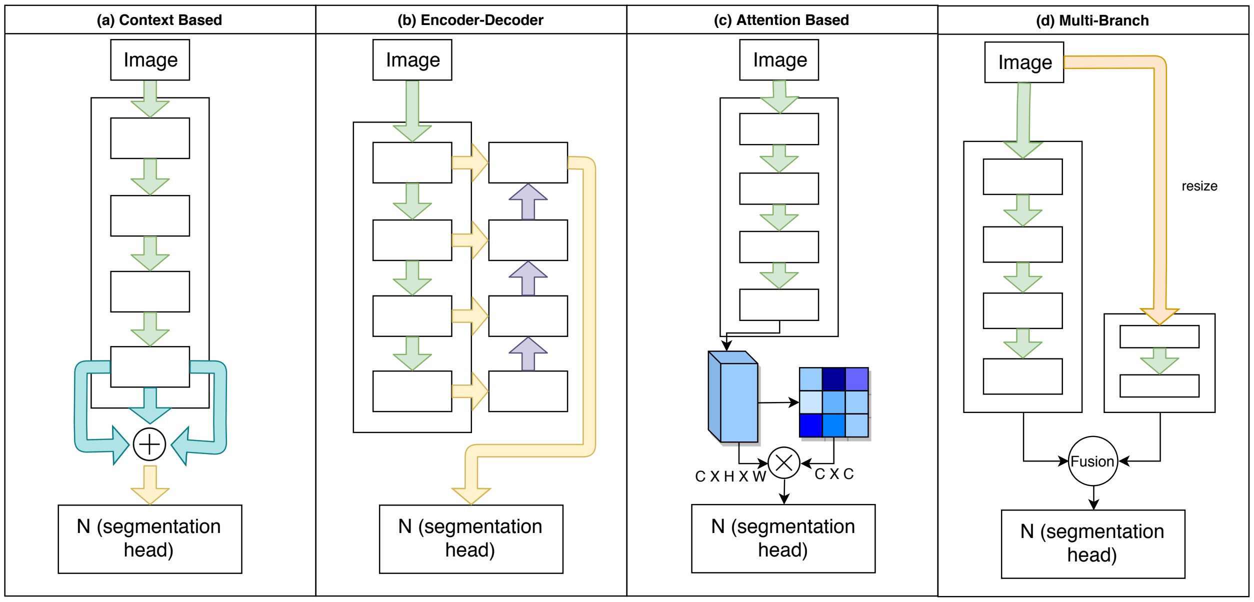


Figure 1: Schematic illustrations of common semantic segmentation architectures. (a) In context-based networks, dilated convolutions with multiple dilation rates are employed in cascade or in parallel to capture a multi-scale context. (b) In encoder-decoder networks, encoder extracts the features of high-level semantic meaning and decoder densify the features learned by the encoder. (c) In attention-based networks, the feature at each position is selectively aggregated by a weighted sum of the features at all positions. This can be done across channels or spatial dimensions. (d) Multi-branch networks are employed to combine semantic segmentation results at multiple resolution levels. The lower resolution branches yield deeper features with reduced resolution and the higher resolution branches learn spatial details.

for the back-propagation. Since training an average deep learning model has a considerable carbon footprint [32], we reduce the training time by 60% with negligible effect on performance by applying progressive resizing for training.

Our main contributions are as follows:

- We propose RGPNet as a general real-time semantic segmentation architecture that obtains deep features with high resolution resulting in improved accuracy and lower latency in a single branch network. It performs competitively in complex environments.
- We introduce an adaptor module to capture multiple levels of abstraction to help in boundary refinement of segments. The adaptor also aids in gradient flow by adding shorter paths.
- Towards green AI, we adopt progressive resizing technique during the training which leads to 60% reduction in training time and the environmental impact. We combat aliasing effect in label map on lower resolutions by employing a modified label relaxation
- We optimize RGPNet for deployment on an edge computing device using TensorRT, a platform for high-performance deep learning inference, resulting in 400% increase in inference speed.
- We report remarkable results on different datasets evaluated on single scale images. RGPNet achieves 80.9%, 69.2%, and 50.2% mIoU with Resnet-101 backbone and 74.1%, 66.9%, and 41.7% mIoU with Resnet-18 backbone on Cityscpes, CamVid and Mapillary, respectively. For a 1024×2048 resolution image, RGPNet obtains 37.4 FPS on NVIDIA GTX2080Ti GPU on the Cityscapes dataset.

2 Related Work

Semantic segmentation lies at the core of computer vision. With the advent of deep learning, Long et al. [20] proposed the seminal fully convolutional network (FCN) with an end-to-end learning approach. However, FCN suffers from the loss of spatial details as it only utilizes high-level features from the last convolutinal layer. Here, we summarize four widely used approaches which have been put forward that increase the feature resolution:

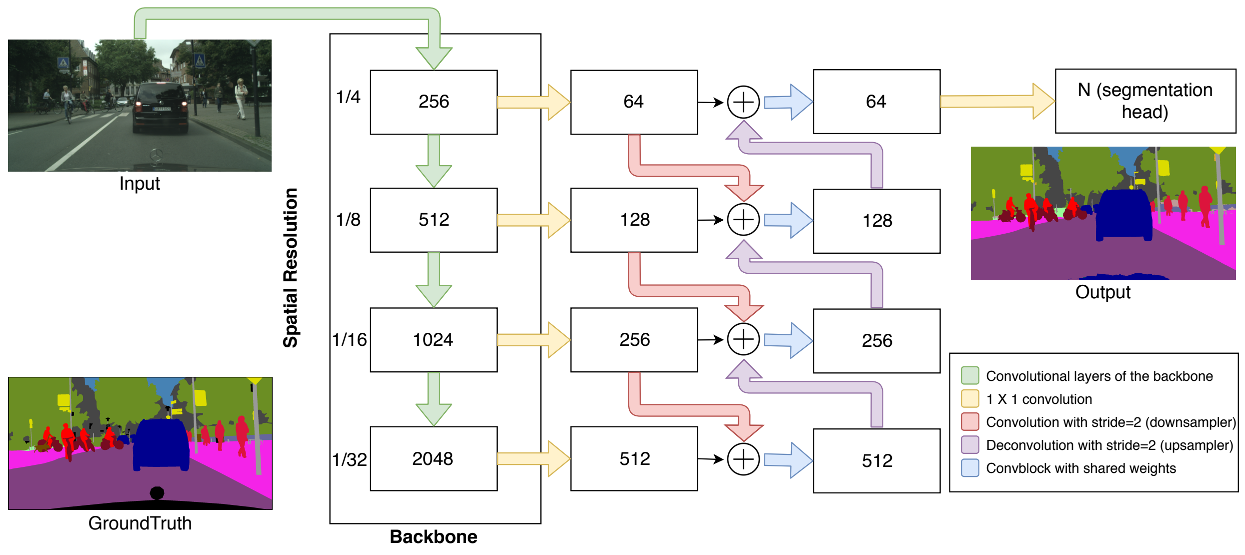

Input
GroundTruth
Spatial Resolution
1/4
1/8
1/16
1/32
256
512
1024
2048
Backbone
64
128
256
512
64
128
256
512
N (segmentation head)
Output
Convolutional layers of the backbone
1 X 1 convolution
Convolution with stride=2 (downsampler)
Deconvolution with stride=2 (upsampler)
Convblock with shared weights

Figure 2: Network schematic diagram of the proposed architecture, RGPNet. Rectangular boxes depict tensor at a given level with number of channels mentioned as their labels. Color coded arrows represent the convolution operations indicated by the legend.

1) Context-based models: To capture the contextual information at multiple scales, DeepLabV2 [4] and DeeplabV3 [5] exploit multiple parallel atrous convolutions with different dilation rates, while PSPNet [44] performs multi-scale spatial pooling operations. Although these methods encode rich contextual information, they can not capture boundary details effectively due to strided convolution or pooling operations [7].

2) Encoder-decoder structure: Several studies entail encode-decoder structure [1, 9, 11, 18, 25, 29, 47]. Encoder extracts global contextual information and decoder recovers the spatial information. Deeplabv3+ [7] utilizes an encoder to extracts rich contextual information in conjunction with a decoder to retrieve the missing object boundary details. However, implementation of dilated convolution at higher dilation rates is computationally intensive making them unsuitable for real-time applications.

3) Attention-based models: Attention mechanisms, which help networks to focus on relevant information and ignore the irrelevant information, have been widely used in different tasks, and gained popularity to boost the performance of semantic segmentation. Wang et al. [37] formalized self-attention by calculating the correlation matrix between each spatial point in the feature maps in video sequences. To capture contextual information, DaNet [10] and OCNet [41] apply a self-attention mechanism. DaNet has dual attention modules on position and channels to integrate local features with their respective global dependencies. OCNet, on the other hand, employs the self-attention mechanism to learn the object context map recording the similarities between all the pixels and the associated pixel. PSANet [45] learns to aggregate contextual information for each individual position via a predicted attention map. Attention based models, however, generally require expensive computation.

4) Multi-Branch models: Another approach to preserve the spatial information is to employ two- or multi-branch approach. The deeper branches extract the contextual information by enlarging receptive fields and shallower branches retain the spatial details. The parallel structure of these networks make them suitable for run time efficient implementations [27, 40, 43]. However, they are mostly applicable to the relatively simpler datasets with fewer number of classes. On the other end, HRNet [34] proposed a model with fully connected links between output maps of different resolutions. This allows the network to generalize better due to multiple paths, acting as ensembles. However, without reduction of spatial dimensions of features, the computational overhead is very high and makes the model no longer feasible for real-time usage.

Building on these observations, we propose a real-time general purpose semantic segmentation architecture that obtains deep features with high resolution resulting in improved accuracy and lower latency in a single branch encoder-decoder network.

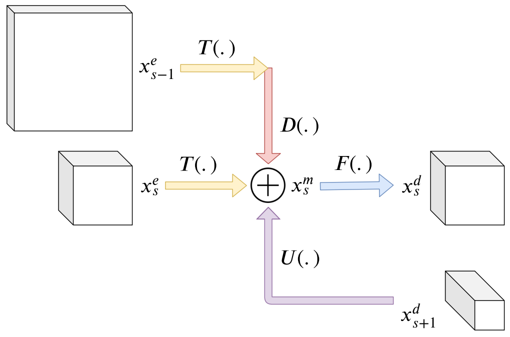


Figure 3: Adaptor module: the adaptor fuses information from multiple abstraction levels; $T(.)$, $D(.)$, and $U(.)$ denote the transfer, downsampling and upsampling functions, respectively. $F(.)$ is the decoder block with shared weights between layers.

3 Proposed Approach

3.1 Structure of RGPNet

RGPNet's design is based on a light-weight asymmetric encoder-decoder structure for fast and efficient inference. It comprises of three components: an encoder which extracts high-level semantic features, a light asymmetric decoder, and an adaptor which links different stages of encoder and decoder. The encoder decreases the resolution and increases the number of feature maps in the deeper layers, thus it extracts more abstract features in deeper layers with enlarged receptive fields. The decoder reconstructs the lost spatial information. The adaptor amalgamates the information from both encoder and decoder allowing the network to preserve and refine the information between multiple levels.

RGPNet architecture is depicted in Figure 2. In a given row of the diagram, all the tensors have the same spatial resolution with number of channels mentioned in the scheme. Four level outputs from the encoder are extracted at different spatial resolutions $1/4$, $1/8$, $1/16$ and $1/32$ with 256, 512, 1024 and 2048 channels, respectively. The number of channels are reduced by a factor of four using 1×1 convolutions followed by batch norm and ReLU activation function at each level. These outputs are then passed through a decoder structure with adaptor in the middle. Finally, segmentation output is extracted from the largest resolution via 1×1 convolution to match the number of channels to segmentation categories.

Adaptor: Adaptor acts as a feature refinement module. The presence of an adaptor precludes the need of a symmetrical encoder-decoder structure. It aggregates the features from three different levels, and intermediates between encoder and decoder (Figure 3). The adaptor function is as below:

$$x^a_s = D(T(x^e_{s-1})) + T(x^e_s) + U(x^d_{s+1}) \tag{1}$$

where superscripts a, e, and d denote *adaptor*, *encoder*, and *decoder* respectively, s represents the spatial level in the network. $D(.)$ and $U(.)$ are downsampling and upsampling functions. Downsampling is carried out by convolution with stride 2 and upsampling is carried out by deconvolution with stride 2 matching spatial resolution as well as the number of channels in the current level. $T(.)$ is a transfer function that reduces the number of output channels from an encoder block and transfers them to the adaptor:

$$T(x^e_s) = \sigma(\omega^a_s \otimes x^e_s + b^a_s) \tag{2}$$

where ω and b are the weight matrix and bias vector, $\otimes$ denotes the convolution operation, and σ denotes the activation function. The decoder contains a modified basic residual block, F, where we use shared weights within the block. The decoder function is as follows:

$$x^d_s = F(x^m_s; \omega^d_s) \tag{3}$$

The adaptor has a number of advantages. First, the adaptor aggregates features from different contextual and spatial levels. Second, it facilitates the flow of gradients from deeper layers to shallower layers by introducing a shorter path. Third, the adaptor allows for utilizing asymmetric design with light-weight decoder. This results in fewer convolution layers, further boosting the flow of gradients. The adaptor, therefore, makes the network suitable for real-time applications as it provides rich semantic information while preserving the spatial information.

3.2 Progressive Resizing with Label Relaxations

Progressive resizing is a technique commonly used in classification to reduce the training time. The training starts with smaller image sizes followed by a progressive increase of size until the final stage of the training is conducted using the original image size. For instance, this technique can theoretically speed up the training time by 16 times per epoch if the image dimensions are decreased by $1/4$ and correspondingly the batch size is increased by a factor of 16 in a single iteration. However, reducing the image size using nearest neighbour interpolation (bi-linear or bi-cubic interpolation are not applicable), introduces noise around the borders of the objects due to aliasing. Note that inaccurate labelling is another source of noise. To reduce effects of boundary artifacts in progressive resizing, inspired by Zhu et al. [46], we use an optimized variant of label relaxation method.

In label relaxation along the borders, instead of maximizing likelihood of a target label, likelihood of union of neighbouring pixel labels is maximized. In our implementation, first one-hot labels are created from the label map followed by max-pool operation with stride 1. This effectively dilates each one-hot label channel transforming it into multi-hot labels along the borders which can then be used to find union of labels along the border pixels. The kernel size of the max pooling controls the width containing pixels being treated as border pixels along the borders. Loss at a given border pixel can be calculated as follows where N is set of border labels:

$$L_{boundary} = -\log \sum_{C \in N} P(C) \tag{4}$$

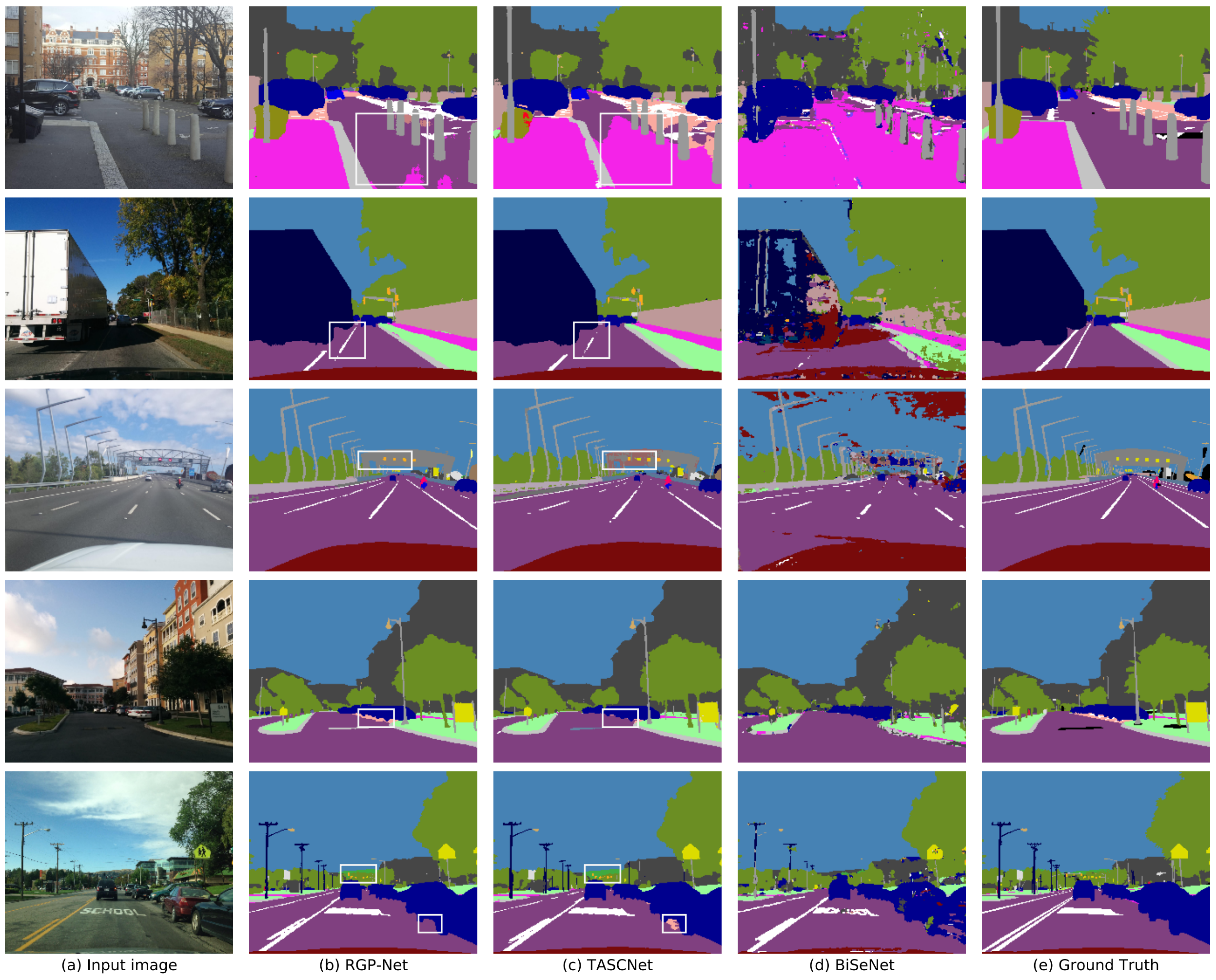


Figure 4: Semantic segmentation results on Mapillary Vistas validation set. The columns correspond to input image, the output of RGPNet, the output of TASCNet, the output of BiSeNet, and the ground-truth annotation. For all methods R101 is used as the backbone. RGPNet mainly improves the results on road and road-related objects' pixels. Best viewed in color and with digital zoom.

4 Experimental Results

We conduct experiments on Mapillary [22] as a highly complex dataset, CamVid [2] and Cityscapes [8] as moderately complex datasets.

Mapillary consists of $20,000$ high-resolution street-level images taken from many different locations around the globe and under varying conditions annotated for 65 categories. The dataset is split up in a training set of $18,000$ images and a validation set of $2,000$ images.

CamVid consists of 701 low-resolution images in 11 classes which are divided into 376/101/233 image sets for training, validation and testing, respectively. Here, we use the same experimental setup as SegNet [1]: 352×480 image resolution for training and inference, 477 images for training and validation, and 233 image as test set.

Cityscapes contains diverse street level images from 50 different city across Germany and France. It contains 30 classes and only 19 classes of them are used for semantic segmentation evaluation. The dataset contains 5000 high quality pixel-level finely annotated images and 20000 coarsely annotated images. The finely annotated 5000 images are divided into $2975/500/1525$ image sets for training, validation and testing. We do not use coarsely annotated data in our experiments.

We implement the RGPNet based on PyTorch framework [24]. For training on both datasets, we employ a polynomial learning rate policy where the initial learning rate is multiplied by $(1 - iter/total\_iter)^{0.9}$ after each iteration. The base learning rate is set to 1×10^{-3}. Momentum and weight decay coefficients are set to 0.9 and 1×10^{-4}, respectively. We train our model with synchronized batch-norm implementation provided by Zhang et al. [42]. Batch size is kept at 12 and trained on two Tesla V100 GPUs. For data augmentation, we apply random cropping and re-scaling with 1024 as crop-size. Image base size is 1536 for Mapillary and 2048 for Cityscapes. Re-scaling is done from range of 0.5 to 2.0 respectively followed by random left-right flipping during training.

As a loss function, we use cross entropy with online hard example mining (OHEM) [38, 41]. OHEM only keeps the sample pixels which are hard for the model to predict in a given iteration. The hard sample pixels are determined by probability threshold θ for the corresponding target class, thus the pixels below the threshold are preserved in the training. To have enough representative of each class in the mini batch, the minimal pixel ratio K is applied. In our experiments, we set $\theta = 0.6$ and $K = 5 \times 10^3$.

4.1 Results on Mapillary

In this section, we evaluate and compare overall performance of RGPNet with other real-time semantic segmentation methods (BiSeNet [40], TASCNet [18], and ShelfNet [47]) on Mapillary validation set. we use different feature extractor backbones ResNet [14] (R101, R50 and R18), Wide-Resnet [39] (WRN38), and HarDNet [3] (HarDNet39D).

Table 1 compares speed (FPS), mIoU and number of parameters on these methods on 16-bit precision computation. RGPNet(R101) achieves 50.2% mIoU which outperforms TASCNet and ShelfNet with a significant margin and lower latency. Although RGPNet(R101) has more parameters than the TASCNet(R101), both speed and mIoU are considerably higher. However, BiSeNet demonstrates poor performance on Mapillary resulting in the lowest mIoU. Using TensorRT, RGPNet (R101 as the encoder) speeds up to 61.9 FPS on full image resolution (Table 7). Our method also achieves impressive results with a lighter encoder (R18 or HarDNet39D) surpassing BiSeNet with a heavy backbone (R101) significantly, 41.7% vs 20.4% mIoU and 54.4 vs 15.5 FPS. Finally, Figure 4 shows some qualitative results obtained by our model compared to TASCNet and BiSeNet.

Table 1: Mapillary Vistas validation set results. The experiments are conducted using 16-bit floating point (FP16) numbers.

Model(backbone)	FPS	mIoU(%)	Params(M)
BiSeNet(R101)	15.5	20.4	50.1
TASCNet(R50)	17.6	46.4	32.8
TASCNet(R101)	13.9	48.8	51.8
ShelfNet(R101)	14.8	49.2	57.7
RGPNet(R101)	**18.2**	**50.2**	**52.2**
RGPNetB(WRN38)	5.72	53.1	215
RGPNet(HarDNet39D)	46.6	42.5	9.4
RGPNet(R18)	**54.4**	**41.7**	**17.8**

4.2 Results on Camvid

In Table 2, we compare overall performance of RGPNet with other real-time semantic segmentation methods (SegNet, FCN [20], FC-DenseNet [15], and FC-HarDNet [3]) on CamVid test set. We find that RGPNet with R18 and R101 backbones obtain 66.9% and 69.2% mIoU with 190 and 68.2 FPS. RGPNet achieves significant increase in mIoU for Car, Traffic Sign, Pole, and Cyclist categories. Overall we observe that our model outperforms the state-of-the-art real-time segmentation models.

Table 2: CamVid test set results. The inference times are calculated on a single NVIDIA TitanV GPU with a single-image batch size.

Model(backbone)	Params(M)	FPS	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Cyclist	mIoU(%)	Pixel Acc.
SegNet	29.5	63.0	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	46.4	62.5
FCN8	135	47.6	77.8	71.0	88.7	76.1	32.7	91.2	41.7	24.4	19.9	72.7	31.0	57.0	88.0
FC-DenseNet56	**1.4**	38.2	77.6	72.0	92.4	73.2	31.8	92.8	37.9	26.2	32.6	79.9	31.1	58.9	88.9
FC-DenseNet103	9.4	20.4	83.0	**77.3**	**93.0**	77.3	43.9	94.5	59.6	37.1	37.8	**82.2**	50.5	66.9	**91.5**
FC-HarDNet68	**1.4**	75.2	80.8	74.4	92.7	76.1	40.6	93.3	47.9	29.3	33.3	78.3	45.7	62.9	90.2
FC-HarDNet84	8.4	34.8	81.4	76.2	92.9	78.3	48.9	**94.6**	61.9	37.9	38.2	80.5	54.0	67.7	91.1
RGPNet(R18)	17.7	**190**	82.6	75.5	91.2	85.1	54.3	94.1	61.5	50.4	36.8	**82.2**	59.8	66.9	90.2
RGPNet(R101)	50.1	68.2	**85.8**	**77.3**	91.2	**87.0**	**62.5**	90.6	**67.6**	**51.4**	**46.8**	70.7	**67.2**	**69.2**	89.9

4.3 Results on Cityscapes

Table 3 shows the comparison between our RGPNet and state-of-the-art real-time (BiSeNet, ICNet [43], FastSCNN [26], and ContextNet [28]) and offline (HRNet [34] and Deeplabv3 [7]) semantic segmentation methods on Cityscapes validation dataset. RGPNet achieves 74.1% mIoU which is slightly lower than BiSeNet 74.8% mIoU. ICNet, ContextNet and FastSCNN achieve lower mIoU. Compared to the heavy offline segmentation methods, RGPNet(R101) not only is the fastest, but also outperforms Deeplabv3, BiSeNet (R101) and is comparable to HRNet.

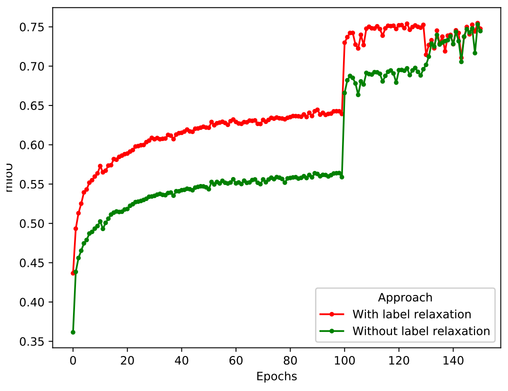
We, therefore, conclude that RGPNet is a real-time general purpose semantic segmentation model that performs competitively a in a wide spectrum of datasets compared to the state-of-the-art semantic segmentation networks designed for specific datasets.

Table 3: Cityscapes validation set result on full resolution image. Numbers with * are taken from respective paper. SS and MS denote single-scale and multi-scale. OOM stands for out-of-memory error. Numbers with † are computed in TensorFlow framework with our in-house implementations.

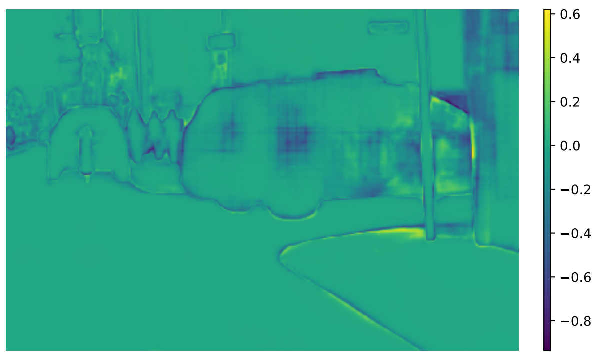
Model		mIoU (%)		FPS
Backbone	Head	SS	MS	
R18	BiSeNet	74.8*	78.6*	40.4
PSPNet50	ICNet	67.7*	-	40.6†
N/A	FastSCNN	68.1	-	43.5†
N/A	ContextNet	60.6	-	37.9†
R18	**RGPNet**	**74.1**	**76.4**	**37.8**
W48	HRNet	81.1*	-	OOM
R101(OS-8)	Deeplabv3	77.82*	79.30*	2.48
R101	BiSeNet	-	80.3*	10.4
R101	**RGPNet**	**80.9**	**81.9**	**10.9**

4.4 Progressive resizing with label relaxation

In order to validate the gain from label relaxation, we compare the result of progressive resizing training with and without label relaxation. In these experiments for the first 100 epochs, the input images are resized by a factor of $1/4$ both in width and height. At the 100^{th} epoch, the image resize factor is set to $1/2$ and, at the 130^{th} epoch, full-sized images are used. With label relaxation, we observe that the model achieves higher mIoU especially at lower resolutions. To further analyze the effect of label relaxation in progressive resizing technique, we illustrate the difference in entropy



(a) Validation mIoU



(b) Differnce of Entropy

Figure 5: (a) Training profiles of progressive resizing with (red) and without (green) label relaxation experiments conducted on Cityscapes validation set. Specially at lower resolutions, label relaxation helps in achieving higher mIoU. (b) Heatmap of difference in entropy between label relaxation and without label relaxation based trained model evaluated on a sample image from validation set. On boundaries of objects, models trained with label relaxation are more confident about the label and hence have lower entropy (blue shades).

between two setups (progressive resizing with and without label relaxation). Figure 5 shows that the model trained with label relaxation is more confident in the predictions around the object boundaries.

Green AI. To examine the energy efficiency, we run two experiments with and without progressive resizing training technique on a single GPU for 15 epochs. In the standard training experiment, we use a full size Cityscapes image 1024×2048. In the progressive resizing training experiment, we start with $1/4$ of image size and then scale up by a factor of 2 at the 10^{th} and the 13^{th} epochs. The speedup factor can theoretically be calculated as $1/16 \times 9/15 + 1/4 \times 3/15 + 3/15 = 0.2875$. Table 4 shows that the training time reduced from 109 minutes to 32 minutes, close to the speedup expected from theoretical calculation. The energy consumed by GPU decreases by an approximate factor of 4 with little to no drop in the performance. Towards green AI, as a result of remarkable gain in energy efficiency we therefore suggest adopting progressive resizing technique with label relaxation for training a semantic segmentation network.

Table 4: Progressive resizing result on energy efficiency. mIoU reported here are from the complete experiment.

Training Scheme	Energy(KJ)	Time	mIoU(%)
Progressive resizing	212	31m 43s	78.8
Full scale	873	108m 44s	80.9

4.5 Ablation study

In this section, we perform an empirical evaluation on the structure of the adaptor module in our design. We remove the downsampling layers which provides information from a higher resolution of encoder to adaptor. Table 5 shows that the performance of our model significantly drops from 50.2% to 46.8% on Mapillary validation set. This indicates that the specific design of adaptor has an important role in feature preserving and refinement in our model.

Table 5: Ablation study result on Mapillary validation set: the effect of downsampling layers which are shown in red in Figure 2 from adaptor.

Method	mIoU(%)
RGPNet	50.2
RGPNet w/o downsampling connections	46.8

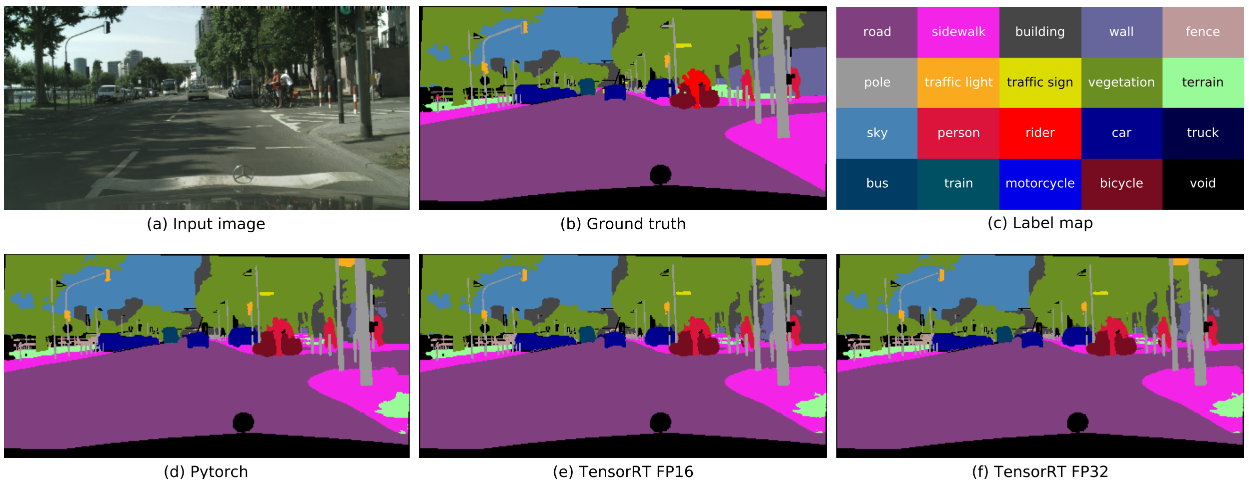


Figure 6: Results obtained by RGPNet on Cityscapes val test. Top row: input image, ground-truth annotation, and label maps. Bottom row: the output of PyTorch model, TensorRT FP16 model, and TensorRT FP32 model. The results show that optimization on TensorRT on half and full precision floating point format does not affect the qualitative outputs.

We also perform an ablation study on common techniques used in the literature on Cityscapes dataset. For Cityscapes training, we adopt a pretrained model on Mapillary dataset by sorting the last layer weights according to mapping between Maplillary and Cityscapes categories. This results in more than 4% boost in mIoU.

Table 6: Ablation study on Cityscapes validation set with RGPNet(R101). CE, OHEM, PM, denote cross-entropy loss, online hard example mining loss, and pretrained model on Mapillary, respectively. MS+Flip stands for multi-scale evaluation with left/right image flip.

CE	OHEM	PM	MS+Flip	mIoU(%)
✓				73.1
✓	✓			76.5
✓	✓	✓		80.9
✓	✓	✓	✓	81.9

4.6 TensorRT

We use TensorRT optimization for RGPNet and evaluate on Nvidia GTX2080Ti and Xavier. RGPNet obtains 79.26% and 79.25% mIoU on Cityscpaes validation with half and full precision floating point format, respectively. The inference speed results for different backbones, two input resolutions using 16-bit and 32-bit floating point numbers are reported in Table 7. RGPNet(R101) using TensorRT on full input resolution leads to a significant increase in speed from 37.8 FPS to 153.4 FPS with 16-bit floating point operations. The speed up with FP16 compared to FP32 is noticeable for all backbones, and two different input resolutions. The results suggest that RGPNet can run high speed on edge computing devices with little or negligible drop in accuracy. A real-world example is provided in Figure 6.

Table 7: RGPNet inference speed using TensorRT on Nvidia GTX2080Ti and Xavier.

Backbone	Nvidia GTX2080Ti				Xavier			
	512×1024		1024×2048		512×1024		1024×2048	
	FP16	FP32	FP16	FP32	FP16	FP32	FP16	FP32
Resnet18	430.2	180.9	153.4	47.2	78.45	24.6	20.8	6.17
Resnet50	265.7	88.8	87.2	24.3	44.6	12.6	11.7	3.17
Resnet101	176.9	58.5	61.9	15.5	30.3	8.14	7.89	2.05

5 Conclusion

In this paper, we proposed a real-time general purpose semantic segmentation network, RGPNet. It incorporates an adaptor module that aggregates features from different abstraction levels and coordinates between encoder and decoder resulting in better gradient flow. Our conceptually simple yet effective model achieves efficient inference speed and accuracy on resource constrained devices in a wide spectrum of complex domains. By employing a modified progressive resizing training scheme, we reduced training time by more than half with no drop in performance, thereby substantially decreasing the carbon footprint. Furthermore, our experiments demonstrate that RGPNet can generate segmentation results in real-time with comparable accuracy to the state-of-the-art non real-time models. This optimal balance of speed and accuracy makes our model suitable for real-time applications such as autonomous driving where the environment is highly dynamic due to the presence of high variability in real world scenarios.

References

[1] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

[2] Brostow, G. J., Fauqueur, J., and Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97.

[3] Chao, P., Kao, C.-Y., Ruan, Y.-S., Huang, C.-H., and Lin, Y.-L. (2019). Hardnet: A low memory traffic network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3552–3561.

[4] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

[5] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

[6] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017b). Rethinking atrous convolution for semantic image segmentation.

[7] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*.

[8] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Ding, H., Jiang, X., Shuai, B., Qun Liu, A., and Wang, G. (2018). Context contrasted feature and gated multi-scale aggregation for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2393–2402.

[10] Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., and Lu, H. (2019a). Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154.

[11] Fu, J., Liu, J., Wang, Y., Zhou, J., Wang, C., and Lu, H. (2019b). Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*.

[12] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[13] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[14] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[15] Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., and Bengio, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–19.

[16] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[17] Lan, X., Zhu, X., and Gong, S. (2018). Person search by multi-scale matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 536–552.

[18] Li, J., Raventos, A., Bhargava, A., Tagawa, T., and Gaidon, A. (2018). Learning to fuse things and stuff.

[19] Li, W., Zhu, X., and Gong, S. (2017). Person re-identification by deep joint learning of multi-loss classification. *arXiv preprint arXiv:1705.04724*.

[20] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[21] Milioto, A., Lottes, P., and Stachniss, C. (2018). Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2229–2235. IEEE.

[22] Neuhold, G., Ollmann, T., Rota Bulo, S., and Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4990–4999.

[23] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.

[24] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

[25] Pohlen, T., Hermans, A., Mathias, M., and Leibe, B. (2017). Full-resolution residual networks for semantic segmentation in street scenes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] Poudel, R. P., Liwicki, S., and Cipolla, R. (2019a). Fast-scnn: fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*.

[27] Poudel, R. P., Liwicki, S., and Cipolla, R. (2019b). Fast-scnn: fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*.

[28] Poudel, R. P. K., Bonde, U., Liwicki, S., and Zach, C. (2018). Contextnet: Exploring context and detail for semantic segmentation in real-time.

[29] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[30] Salehi, S. S. M., Hashemi, S. R., Velasco-Annis, C., Ouaalam, A., Estroff, J. A., Erdogmus, D., Warfield, S. K., and Gholipour, A. (2018). Real-time automatic fetal brain extraction in fetal mri by deep learning. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 720–724. IEEE.

[31] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[32] Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

[33] Su, Y.-H., Huang, K., and Hannaford, B. (2018). Real-time vision-based surgical tool segmentation with robot kinematics prior. In *2018 International Symposium on Medical Robotics (ISMR)*, pages 1–6. IEEE.

[34] Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., and Wang, J. (2019). High-resolution representations for labeling pixels and regions. *CoRR*, abs/1904.04514.

[35] Sun, S., Pang, J., Shi, J., Yi, S., and Ouyang, W. (2018). Fishnet: A versatile backbone for image, region, and pixel level prediction. In *Advances in Neural Information Processing Systems*, pages 754–764.

[36] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[37] Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803.

[38] Wu, Z., Shen, C., and Hengel, A. v. d. (2016). High-performance semantic segmentation using very deep fully convolutional networks. *arXiv preprint arXiv:1604.04339*.

[39] Wu, Z., Shen, C., and van den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133.

[40] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., and Sang, N. (2018). Bisenet: Bilateral segmentation network for real-time semantic segmentation. *Lecture Notes in Computer Science*, page 334–349.

[41] Yuan, Y. and Wang, J. (2018). Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*.

[42] Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., and Agrawal, A. (2018). Context encoding for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] Zhao, H., Qi, X., Shen, X., Shi, J., and Jia, J. (2018a). Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420.

[44] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[45] Zhao, H., Zhang, Y., Liu, S., Shi, J., Change Loy, C., Lin, D., and Jia, J. (2018b). Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283.

[46] Zhu, Y., Sapra, K., Reda, F. A., Shih, K. J., Newsam, S., Tao, A., and Catanzaro, B. (2018). Improving semantic segmentation via video propagation and label relaxation.

[47] Zhuang, J. and Yang, J. (2018). Shelfnet for real-time semantic segmentation. *arXiv preprint arXiv:1811.11254*.