# RGPNet: A Real-Time General Purpose Semantic Segmentation

Elahe Arani[*], Shabbir Marzban[*], Andrei Pata, and Bahram Zonooz
Advanced Research Lab, NavInfo Europe, Eindhoven, The Netherlands
{elahe.arani, shabbir.marzban, andrei.pata}@navinfo.eu, bahram.zonooz@gmail.com

## Abstract

*We propose a real-time general purpose semantic segmentation architecture, 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 experiments demonstrate that RGPNet can generate segmentation results in real-time with comparable accuracy to the state-of-the-art non-real-time heavy models. Moreover, towards green AI, we show that using an optimized label-relaxation technique with progressive resizing can reduce the training time by up to 60% while preserving the 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, 32, 37]. 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, 24, 31, 34]. 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 [30, 1, 36], a two or multi-branch architecture [27, 44, 41] or dilated convolutions [4, 5, 6, 45] 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 short paths for the back-propagation. Since training an average deep learning model has a considerable carbon footprint [33], 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 short 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

[*] Equal contribution.

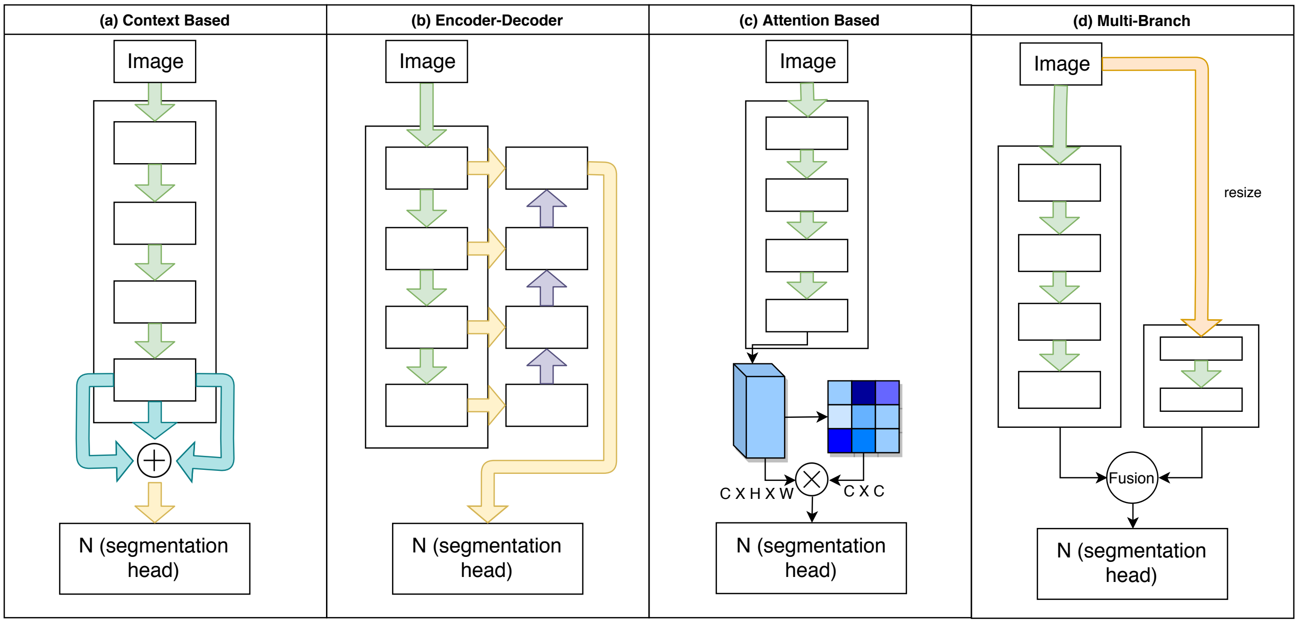


Figure 1. Schematic illustrations of common semantic segmentation architectures. (a) In context-based networks, dilated convolutions with multiple dilation rates are employed in cascade or in parallel to capture a multi-scale context. (b) In encoder-decoder networks, encoder extracts the features of 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.

- We report 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 Cityscapes, CamVid and Mapillary, respectively.

- For a $1024 \times 2048$ resolution image, RGPNet(Resnet101) obtains 10.9 FPS in Pytorch and 15.5 FPS in TensorRT on NVIDIA RTX2080Ti GPU on the Cityscapes dataset whereas RGPNet(Resnet18) obtains 37.8 FPS and 47.2 FPS respectively under the same setting.

## 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 convolutional layer. Here, we summarize four widely used approaches which have been put forward that increase the feature resolution:

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 [45] 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 [30, 1, 26, 48, 18, 9, 11]. 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*. [38] 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 [42] apply a self-attention mechanism. DaNet has dual attention modules on position and channels to integrate local features with their respective global dependen-

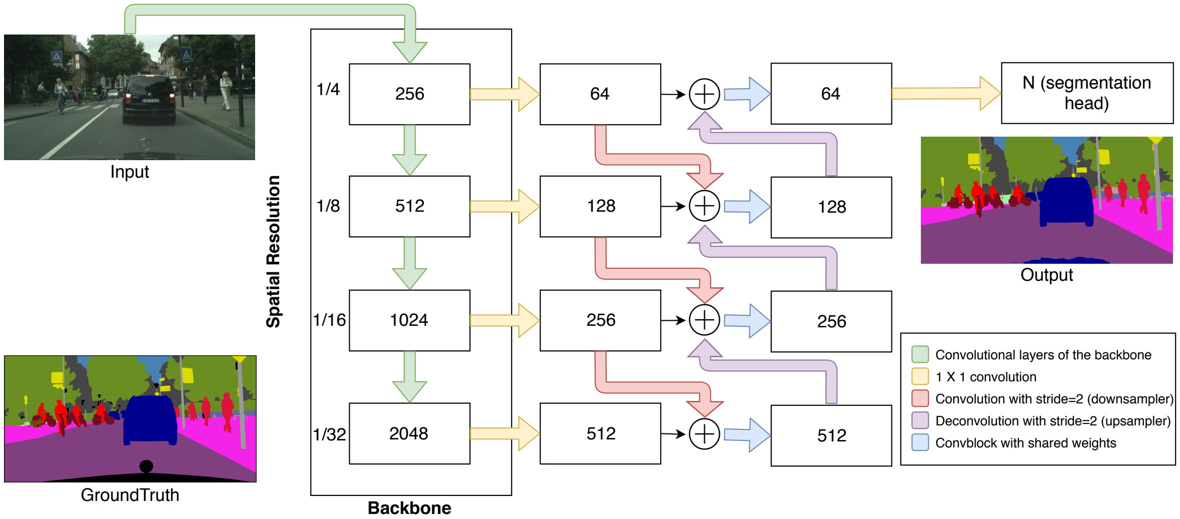

Input
GroundTruth
Spatial Resolution
1/4
1/8
1/16
1/32
256
512
1024
2048
Backbone
64
128
256
512
64
128
256
512
N (segmentation head)
Output
🟩 Convolutional layers of the backbone
🟨 1 X 1 convolution
🟥 Convolution with stride=2 (downsampler)
🟪 Deconvolution with stride=2 (upsampler)
🟦 Convblock with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

cies. 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 [46] 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 [41, 44, 28, 23]. However, they are mostly applicable to the relatively simpler datasets with fewer number of classes. On the other end, HRNet [35] 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.

## 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 the 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 $1 \times 1$ 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 $1 \times 1$ 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

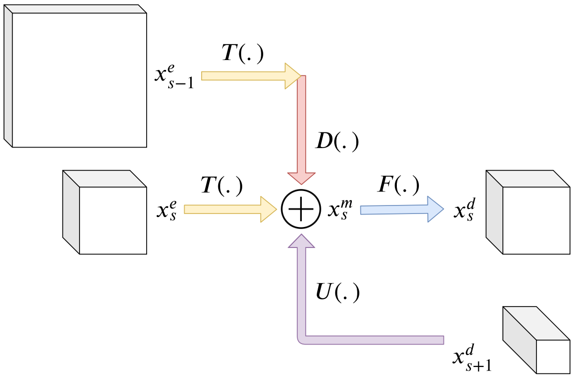


Figure 3. Adaptor module: the adaptor fuses information from multiple abstraction levels; $T(.)$, $D(.)$, and $U(.)$ denote the transfer, downsampling and upsampling functions, respectively. $F(.)$ is the decoder block with shared weights between layers.

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:

$$T(x_s^e) = \sigma(\omega_s^a \otimes x_s^e + b_s^a) \tag{2}$$

where $\omega$ and $b$ are the weight matrix and bias vector, $\otimes$ denotes the convolution operation, and $\sigma$ 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:

$$x_s^d = F(x_s^m; \omega_s^d) \tag{3}$$

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, applying progressive resizing for semantic segmentation is more challenging as the resizing method should be applied to images and their corresponding label maps. Bi-linear or bi-cubic interpolation cannot be applied to label maps as they exist in integer space and these methods will result in float values for labels. On the other hand, nearest-neighbor interpolation for resizing introduces noise in the label maps around the borders of the objects due to aliasing. To reduce the effects of boundary artifacts in progressive resizing for label maps, inspired by Zhu *et al.* [47], we propose an optimized variant of the label relaxation method.

In cross-entropy loss function, the negative log-likelihood of softmax probability for a given label is maximized. In contrast, label-relaxation is a custom loss function where the negative log-likelihood of softmax probabilities for a given label as well as bordering pixel labels is maximized. This is established by taking the sum of softmax probabilities mentioned, before applying the negative log-likelihood. We identify the border pixels as those which have more than one unique label in the window with kernel size $k$ centered on it. The loss at a given border pixel can be calculated as follows where N is a set of border labels:

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

To apply label relaxation efficiently, 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 enables optimized selection of border pixels along with their corresponding labels. Border pixels are usually in minority; for instance, in the Cityscapes dataset, on average, only $2.4\%$ of total pixels are border pixels. This loss function is only applied to border pixels whereas normal cross-entropy loss is applied to the rest of the pixels.

## 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

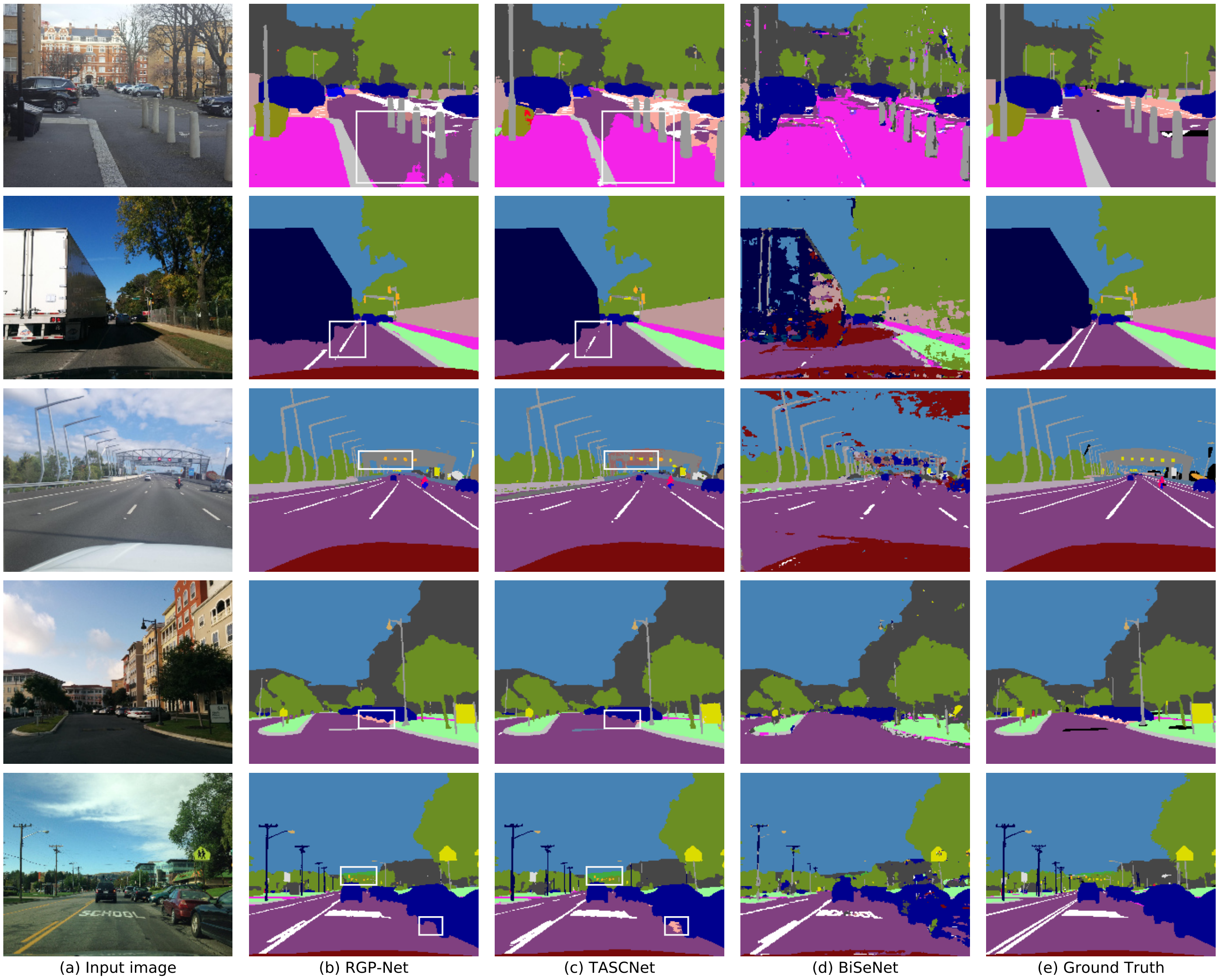


Figure 4. Semantic segmentation results on Mapillary Vistas validation set. The columns correspond to input image, the output of RGPNet, the output of TASCNet, the output of BiSeNet, and the ground-truth annotation. For all methods R101 is used as the backbone. RGPNet mainly improves the results on road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

the same experimental setup as SegNet [1]: $352 \times 480$ 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 cities. 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 [25]. 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 $1 \times 10^{-3}$. Momentum and weight decay coefficients are set to 0.9 and $1 \times 10^{-4}$, respectively. We train our model with synchronized batchnorm implementation provided by Zhang *et al.* [43]. Batch size is kept at 12 and trained on two Tesla V100 GPUs. For data augmentation, we apply random cropping and rescaling 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) [39, 42]. 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 $\theta$ for the corresponding target class, thus the pixels below the threshold are preserved in the

Table 1. CamVid test set results calculated on $352 \times 480$ image resolution. 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 |

training. To have enough representative of each class in the mini batch, the minimal pixel ratio $M$ is applied. In our experiments, we set $\theta = 0.6$ and $M = 5 \times 10^3$.

### 4.1. Results on Mapillary

In this section, we evaluate and compare overall performance of RGPNet with other real-time semantic segmentation methods (BiSeNet [41], TASCNet [18], and ShelfNet [48]) on Mapillary validation set. we use different feature extractor backbones ResNet [14] (R101, R50 and R18), Wide-Resnet [40] (WRN38), and HarDNet [3] (HarDNet39D).

Table 2 compares speed (FPS), mIoU and number of parameters on these methods on 32-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 5). 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.

### 4.2. Results on Camvid

In Table 1, 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. Mapillary Vistas validation set. Inference speed is calculated on $1024 \times 2048$ image resolution.

| Model(backbone) | FPS | mIoU(%) | Params(M) |
|---|---|---|---|
| BiSeNet(R101) | 9.27 | 20.4 | 50.1 |
| TASCNet(R50) | 11.9 | 46.4 | 32.8 |
| TASCNet(R101) | 8.84 | 48.8 | 51.8 |
| ShelfNet(R101) | 9.11 | 49.2 | 57.7 |
| **RGPNet(R101)** | **10.8** | **50.2** | **52.2** |
| RGPNetB(WRN38) | 3.37 | 53.1 | 215 |
| RGPNet(HarDNet39D) | 34.7 | 42.5 | 9.4 |
| **RGPNet(R18)** | **35.7** | **41.7** | **17.8** |

### 4.3. Results on Cityscapes

Table 3 shows the comparison between our RGPNet and state-of-the-art real-time (BiSeNet, ICNet [44], FastSCNN [27], and ContextNet [29]) and offline (HRNet [35] 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 in a wide spectrum of datasets compared to the state-of-the-art semantic segmentation networks designed for specific datasets.

### 4.4. Progressive resizing with label relaxation

Here, 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. To analyze the effect of label re-

Table 3. Cityscapes validation set result on $1024 \times 2048$ 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 which are better than originally reported in respective paper.

| 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 | SwiftNet | 75.4* | - | 56.3 |
| **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** |

laxation in progressive resizing technique, we illustrate the difference in entropy 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 the experiments with and without progressive resizing training technique and label relaxation on a single GPU for 15 epochs. In the standard training experiment, we use a full size Cityscapes image $1024 \times 2048$. 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. Note that, inclusion of our optimized label relaxation causes a minute increase in the energy consumption (less than 10 KJ) and time (4 minutes more). The energy consumed by GPU decreases by an approximate factor of 4 when compared to full scale experiment with slight 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. PR and LR stand for progressive resizing and label relaxation, respectively. mIoU reported here are from the complete experiment.

| Training Scheme | Energy(KJ) | Time | mIoU(%) |
|---|---|---|---|
| PR w/o LR | 203 | 27m 37s | 78.3 |
| PR with LR | 212 | 31m 43s | 78.8 |
| Full scale | 873 | 108m 44s | 80.9 |

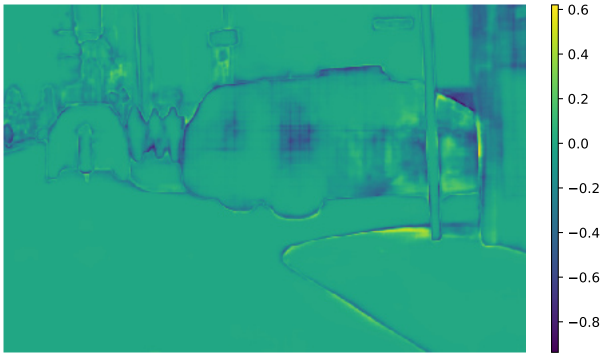


Figure 5. Heatmap of difference in entropy between label relaxation and without label relaxation based trained model evaluated on a sample image from validation set. On boundaries of objects, models trained with label relaxation are more confident about the label and hence have lower entropy (blue shades).

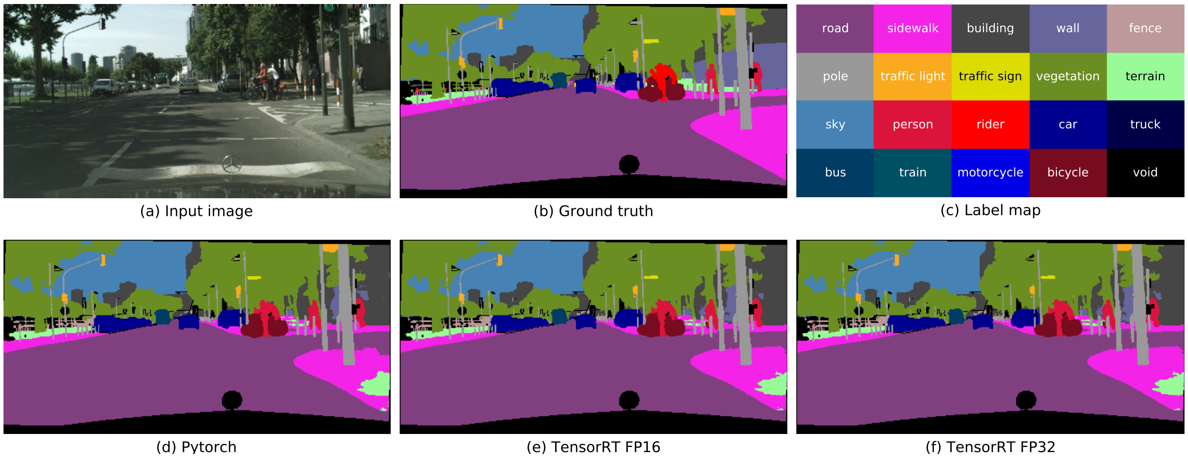
## 4.5. Ablation study

In this section, we perform an empirical evaluation on the structure of the adaptor module in our design. We show the significance of the downsampling layers which provides information from a higher resolution of encoder to adaptor. Table 6 shows that the performance of our model significantly drops from 50.2% to 46.8% on Mapillary validation set when the downsampling layers are removed. This indicates that the specific design of adaptor has an important role in feature preserving and refinement in our model. We show similar effect in in Table 7 which showcases results on Cityscapes dataset where adding downsampling layers result in boost in mIoU with Resnet101 backbone.

We perform an ablation study on components of our training framework on Cityscapes dataset. These techniques are namely; cross entropy (CE), addition of downsampling layers (DS), cross entropy with online hard example mining (OHEM) and using pretrain weights from Mapillary (PM). For the last technique mentioned, 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 4.2% and 2.4% boost in mIoU with Resnet101 and Resnet18 backbones respectively while evaluating on multiple scales. As illustrated in Table 7, all these components contribute significantly in improving the final performance.

## 4.6. TensorRT

We use TensorRT for RGPNet and evaluate on Nvidia RTX2080Ti 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 5. RGPNet(R18) using TensorRT on full input res-



(a) Input image (b) Ground truth (c) Label map

(d) Pytorch (e) TensorRT FP16 (f) TensorRT FP32

Figure 6. Results obtained by RGPNet on Cityscapes validation set on 1024 × 2048 image resolution. 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.

Table 5. RGPNet inference speed (FPS) using TensorRT on Nvidia RTX2080Ti and Xavier evaluated on half and full resolution images from Cityscapes dataset. Comparison between floating point 16 bit (FP16) and 32 bit (FP32) computation is also shown.

| Backbone | Nvidia RTX2080Ti | | | | Xavier | | | |
|---|---|---|---|---|---|---|---|---|
| | 512 × 1024 | | 1024 × 2048 | | 512 × 1024 | | 1024 × 2048 | |
| | FP16 | FP32 | FP16 | FP32 | FP16 | FP32 | FP16 | FP32 |
| R18 | 430.2 | 180.9 | 153.4 | 47.2 | 78.45 | 24.6 | 20.8 | 6.17 |
| R50 | 265.7 | 88.8 | 87.2 | 24.3 | 44.6 | 12.6 | 11.7 | 3.17 |
| R101 | 176.9 | 58.5 | 61.9 | 15.5 | 30.3 | 8.14 | 7.89 | 2.05 |

Table 6. Ablation study on Mapillary validation set: it highlights the effect of downsampling layers which are shown in red in Figure 2 from adaptor. MS+F stands for multi-scale evaluation with left/right image flip.

| Method | DS | MS+F mIoU(%) |
|---|---|---|
| RGPNet(R101) | | 46.8 |
| RGPNet(R101) | ✓ | 50.2 |

Table 7. Ablation study on Cityscapes validation set with RGPNet on 1024 × 2048 image resolution. CE, DS, OHEM, PM denote cross-entropy loss, down-sampling connections, online hard example mining loss, and pretrained model on Mapillary, respectively. MS+F stands for multi-scale evaluation with left/right image flip.

| CE | DS | OHEM | PM | R101 | | R18 | |
|---|---|---|---|---|---|---|---|
| | | | | SS | MS+F | SS | MS+F |
| ✓ | | | | 73.0 | 74.6 | 69.1 | 71.2 |
| ✓ | ✓ | | | 73.1 | 75.5 | 69.0 | 71.3 |
| ✓ | ✓ | ✓ | | 76.5 | 77.7 | 71.9 | 74.0 |
| ✓ | ✓ | ✓ | ✓ | 80.9 | 81.9 | 74.1 | 76.4 |

olution 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.

## 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 a 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 an optimized progressive resizing training scheme, we reduced training time by more than half with a small 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] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

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

[3] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3552–3561, 2019.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 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, 2017.

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[9] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. 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, 2018.

[10] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[11] Jun Fu, Jing Liu, Yuhang Wang, Jin Zhou, Changyong Wang, and Hanqing Lu. Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*, 2019.

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

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[15] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. 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, 2017.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[18] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff, 2018.

[19] Wei Li, Xiatian Zhu, and Shaogang Gong. Person re-identification by deep joint learning of multi-loss classification. *arXiv preprint arXiv:1705.04724*, 2017.

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

[21] Andres Milioto, Philipp Lottes, and Cyrill Stachniss. 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, 2018.

[22] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4990–4999, 2017.

[23] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12607–12616, 2019.

[24] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *In NeurIPS Workshop*, 2017.

[26] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[27] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019.

[28] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. Fast-scnn: fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019.

[29] Rudra P K Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. Contextnet: Exploring context and detail for semantic segmentation in real-time, 2018.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[31] Seyed Sadegh Mohseni Salehi, Seyed Raein Hashemi, Clemente Velasco-Annis, Abdelhakim Ouaalam, Judy A Estroff, Deniz Erdogmus, Simon K Warfield, and Ali Gholipour. 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, 2018.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[34] Yun-Hsuan Su, Kevin Huang, and Blake Hannaford. Real-time vision-based surgical tool segmentation with robot kinematics prior. In *2018 International Symposium on Medical Robotics (ISMR)*, pages 1–6. IEEE, 2018.

[35] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. *CoRR*, abs/1904.04514, 2019.

[36] Shuyang Sun, Jiangmiao Pang, Jianping Shi, Shuai Yi, and Wanli Ouyang. Fishnet: A versatile backbone for image, region, and pixel level prediction. In *Advances in Neural Information Processing Systems*, pages 754–764, 2018.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[39] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. High-performance semantic segmentation using very deep fully convolutional networks. *arXiv preprint arXiv:1604.04339*, 2016.

[40] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, Jun 2019.

[41] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation, 2018.

[42] Yuhui Yuan and Jingdong Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018.

[43] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[44] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.

[45] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[46] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018.

[47] Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation, 2018.

[48] Juntang Zhuang and Junlin Yang. Shelfnet for real-time semantic segmentation. *arXiv preprint arXiv:1811.11254*, 2018.