Guided Upsampling Network for Real-Time Semantic Segmentation

Davide Mazzini
davide.mazzini@unimib.it

Department of Informatics, Systems and Communication
University of Milano-Bicocca
viale Sarca 336 Milano, Italy

Abstract

Semantic segmentation architectures are mainly built upon an encoder-decoder structure. These models perform subsequent downsampling operations in the encoder. Since operations on high-resolution activation maps are computationally expensive, usually the decoder produces output segmentation maps by upsampling with parameters-free operators like bilinear or nearest-neighbor. We propose a Neural Network named Guided Upsampling Network which consists of a multiresolution architecture that jointly exploits high-resolution and large context information. Then we introduce a new module named Guided Upsampling Module (GUM) that enriches upsampling operators by introducing a learnable transformation for semantic maps. It can be plugged into any existing encoder-decoder architecture with little modifications and low additional computation cost. We show with quantitative and qualitative experiments how our network benefits from the use of GUM module. A comprehensive set of experiments on the publicly available Cityscapes dataset demonstrates that Guided Upsampling Network can efficiently process high-resolution images in real-time while attaining state-of-the art performances.

1 Introduction

Most of the current state-of-the-art architectures for image segmentation rely on an encoder-decoder structure to obtain high-resolution predictions and, at the same time, to exploit large context information. One way to increase network receptive fields is to perform downsampling operations like pooling or convolutions with large stride. Reduction of spatial resolution is twice beneficial because it also lightens the computational burden. Even state-of-the-art architectures that make use of dilated convolutions [6, 23, 25], employ some downsampling operators in order to maintain the computation feasible. Semantic maps are usually predicted at $1/8$ or $1/16$ of the target resolution and then they are upsampled using nearest neighbor or bilinear interpolation.

1.1 Our focus and contribution

We focus on Semantic Segmentation of street scenes for automotive applications where a model needs to be run continuously on vehicles to take fast decisions in response to environmental events. For this reason, our design choices are the result of a trade-off between

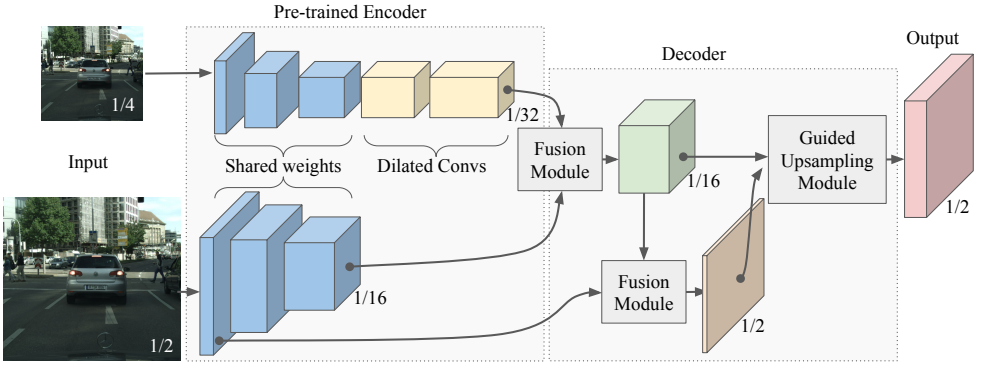


Figure 1: Guided Upsampling Network (GUN) architecture. Two branches with partially shared weights extract fine and coarse features. Signals are merged by a *fusion module*. Output is produced by Guided Upsampling Module (GUM) which exploits high-resolution features from earliest layers and complex features from deepest ones.

processing speed and accuracy. Our work focuses on a fast architecture with a lightweight decoder that makes use of a more effective upsampling operator. Our contributions are the following:

- We developed a novel multi-resolution network architecture named Guided Upsampling Network, presented in Section 3 that is able to achieve high-quality predictions without sacrificing speed. Our system can process a 512x1024 resolution image on a single GPU at 33 FPS while attaining 70.4% IoU on the cityscapes test dataset.
- We designed our network in an incremental way outlining pros and cons of every choice and we included all crucial implementation details in Section 3.1 to make our experiments easily repeatable.
- We designed a novel module named GUM (Guided Upsampling Module, introduced in Section 4) to efficiently exploit high-resolution clues during upsampling.

1.2 Related works

[7, 8, 17] represent the pioneer works that employed CNNs for semantic segmentation. FCN [15] laid the foundations for modern architectures where CNNs are employed in a fully-convolutional way. Authors used a pre-trained encoder together with a simple decoder module that takes advantage of skip-connections from lower layers to exploit high-resolution feature maps. They obtained a significant improvement both in terms of accuracy and efficiency. DeepLab [2] made use of Dilated Convolutions [22] to increase the receptive field of inner layers without increasing the overall number of parameters. After the introduction of Residual Networks (Resnets) [9] most methods employed a very deep Resnet as encoder *e.g.* DeepLabv2 [4] Resnet38 [21] FRRN [18], pushing forward the performance boundary on semantic segmentation task. PSPNet [25] and DeepLabv3 [5] introduced context layers in order to expand the theoretical receptive field of inner layers. All these methods attain high accuracy on different benchmarks but at high computational costs.

Efficiency-oriented architectures. ENet authors [16] headed towards a high-speed architecture, dramatically raising model efficiency, but sacrificing accuracy. SegNet [1] introduced an efficient way to exploit high-resolution information by saving max-pooling indices from the encoder and using them during upsampling. ICNet [24] design is based on a three branches architecture exploiting deep supervision for training. ERFNet [19] implements an efficient Residual Factorized Convolution layer in order to achieve high a accuracy while being particularly efficient.

2 Dataset and evaluation metrics

All the experiments presented in this work have been performed on Cityscapes [6]. It is a dataset of urban scenes images with semantic pixelwise annotations. It consists of 5000 finely annotated high-resolution images (2048x1024) of which 2975, 500, and 1525 belong to train, validation and test sets respectively. Annotations include 30 different object classes but only 19 are used to train and evaluate models. Adopted evaluation metrics are *mean of class-wise Intersection over Union (mIoU)* and *Frame Per Second (FPS)*, defined as the inverse of time needed for our network to perform a single forward pass. FPS reported in the following sections are estimated on a single Titan Xp GPU.

3 Network design

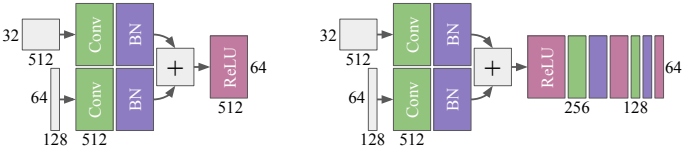
In this section we describe in details our network architecture. Most works in literature expose the final model followed by an ablation study. This is motivated by an implicit inductive prior towards simpler models, *i.e.* simpler is better. Even though we agree with this line of thought we designed our experiments following a different path: by incremental steps. We started from a baseline model and incrementally added single features analyzing benefits and disadvantages. Our network architecture, based on a fully-convolutional encoder-decoder, is presented in details in the following subsections.

Input downsampling A naive way to speed up inference process in real-time applications is to subsample the the input image. This comes at a price. Loss of fine details hurts performance because borders between classes and fine texture information are lost. We investigated a trade-off between system speed and accuracy. We used a DRN-D-22 model [23] pre-trained on Imagenet as encoder and a simple bilinear upsampling as decoder. First column of Table 1 shows the mIoU of the baseline model without any subsampling. In the second column the same model is trained and evaluated with input images subsampled by factor 4. Model speed increases from 6.7 FPS to 50.6 which is far beyond real-time but, as expected, there is a big (8%) performance drop.

Multiresolution encoder As a second experiment we designed a multi-resolution architecture as a good compromise to speed up the system without sacrificing its discriminative power. Our encoder consists of two branches: a low-resolution branch which is composed of all the layers of a Dilated Residual Network 22 type D (DRN-D-22) [23] with the exception of the last two. A medium-resolution branch with only the first layers of the DRN-D-22 before dilated convolutions. The main idea is to induce the first branch to extract large context features while inducing the second to extract more local features that will help to recover fine details during decoding. We experimented 3 different encoder configurations. The first named *enc24* in Table 1 consists of two branches that process input images with

Encoder	baseline	enc4	enc24	**enc24shared**	enc124shared
Multiresolution			✓	✓	✓
Shared Parameters				✓	✓
Subsampling factor	1	4	2 + 4	**2 + 4**	1 + 2 + 4
mIoU (%)	65.5	57.5	61.5	**63.0**	64.2
FPS	6.7	50.6	38.7	**38.7**	24.9

Table 1: Performance on Cityscapes validation set and speed (FPS) of four encoder architectures. *baseline* is a full-resolution network. *enc4* is trained and evaluated with downsampled input. *enc24* and *enc124* means 2 and 3 branches with subsampling factors 2,4 and 1,2,4 respectively. *shared* means that weights are partially shared between branches. In bold the configuration adopted in the final model.



(a) base sum (b) postproc sum

Figure 2: Two *fusion module* configurations. Both exploit addiction as merge strategy. *postproc sum* performs a dimensionality reduction.

sub-sampling factors 2 and 4 with the structure defined above. The second configuration named *enc24shared* is similar to the first. The only difference is weight sharing between the two branches. Results in Table 1 show that the network with shared branches achieve better performance. We argue that, by reducing the number of network parameters, weight sharing between branches, induces an implicit form of regularization. For this reason we used this configuration as base encoder for the next experiments. In the third configuration named *enc124shared* in Table 1 we added a further branch to elaborate full-resolution image. This indeed brought some performance improvements but we decided to discard this configuration because operations at full resolution are computationally too heavy and the whole system would slow down below the real-time threshold (30FPS). To train and evaluate the different encoder designs in Table 1 we fixed the decoder architecture to a configuration which is referred in Subsection 3 as *baseline*. Figure 1 depicts the second encoder design *enc24shared*. Others have been omitted for space reasons but can be intuitively deduced.

Fusion module. It is the first part of our decoder. It joins information flows coming from the two encoder branches extracted at multiple resolutions. Input from low-resolution

Fusion Module	base sum	base concat	**postproc sum**	postproc concat
Sum	✓		✓	
Concat		✓		✓
Postprocessing Step			✓	✓
mIoU (%)	63.0	63.5	**65.8**	64.2
FPS	38.7	37.8	**37.3**	36.4

Table 2: mIoU on Cityscapes val set and FPS for different *fusion modules*. Differences are: signal summation or concatenation and presence of a post-processing step. In bold the configuration adopted in the final model.

Transformation	baseline	color jitter	lighting jitter	**random scale**
mIoU (%)	65.8	62.6	64.2	**67.5**

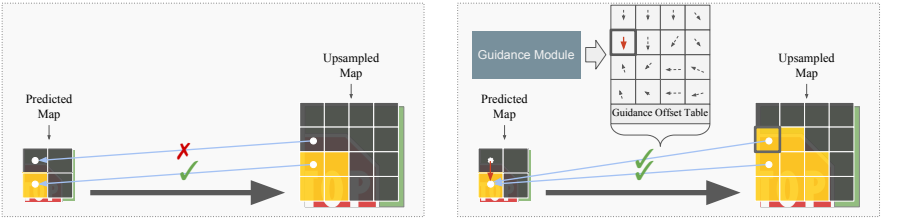
Table 3: mIoU on Cityscapes validation set with different data augmentation techniques used during training. In bold the configuration adopted in the final model.

branch is up-sampled to match the spatial size of signal coming from the medium-resolution branch. Input coming from medium-resolution branch is expanded from 128 to 512 channels to match the number of features of the first branch. Then multi-resolution signals are merged and further processed. In Table 2 are reported experimental results of four different designs. We experimented channel concatenation and addiction as merge strategies for signals coming from the two branches, named *concat* and *sum* respectively. We further investigated if the network benefits from feeding the final classification layer directly with the signal after the merge operation (*base* in Table 2), or if a dimensionality reduction brings improvements (*postproc*). From experimental results shown in Table 2 both mIoU and speed take advantage of the post-processing step. The model is empowered by adding more convolutions and non-linearities and the final upsampling operations are applied to a smaller feature space. Figure 2 depicts two different configurations: *base sum* and *postproc sum*, both with addiction merge strategy, without and with the post-processing step. Fusion modules with *concat* as merge strategy have a similar structure.

3.1 Training recipes

In this section we expose our *training recipes*: some considerations about hyper-parameters and their values used to train our models plus a small paragraph on synthetic data augmentation. For all experiments in this paper we trained the network with SGD plus momentum. Following [23] we set learning rate to 0.001 and trained every model for at least 250 epochs. We adopted a *step* learning rate policy. The initial value is decreased every 100 epochs by a order of magnitude. We also tried different base learning rates and *poly* learning rate policy from [2] but we obtained better results with our baseline configuration. We found out that batch size is a very sensitive parameter affecting the final accuracy. After experimenting with different values we set it to 8. In contrast to what pointed out in [3], increasing the batch size, in our case, hurts performance. Batch size affect performance because of intra-batch dependencies introduced by Batch Normalization layers. We argue that, in our case, the higher stochasticity introduced by intra-batch dependencies acts as regularizer, thus effectively improving the final network performance.

Synthetic data augmentation. Considering the low amount of data used to train our network *i.e.* 2970 fine-annotated images from Cityscapes dataset, we decided to investigate some well-known data augmentation techniques. The application of these techniques is almost cost-free in terms of computational resources. They do not increase processing time during inference and they can be applied as a CPU pre-processing step during training. This is in line with the research direction of this work which goal is to push forward accuracy while maintaining a real-time inference speed. Since our system is supposed to work with outdoor scenes and thus dealing with a high variability of lighting conditions we experimented the use of some light transformations. Color Jitter consists in modifying image brightness, saturation and contrast in random-order. Lighting Jitter is a PCA based noise jittering from [13], we used $\sigma = 0.1$ as standard deviation to generate random noise. We



(a) Nearest Neighbor (b) Guided Upsampling

Figure 3: Guided Upsampling Module (GUM). The Guidance Module produces a Guidance Offset Table which steers the upsampling process.

also experimented a geometric transform: rescaling image with a scale factor between 0.5 and 2, borrowing values from [23]. Table 3 shows the results of applying data augmentation techniques described in this section. Only *random scale* brought some improvements, thus we decided to include it in our training pipeline for the next experiments.

4 Guided upsampling module

In this section we introduce Guided Upsampling Module (GUM). It is born from the intuition that generating a semantic map by predicting independently every single pixel is quite inefficient. It is a matter of fact that most algorithms that perform semantic segmentation do not predict full resolution maps [5, 15, 23, 24, 25]. They produce a low-resolution map that is upsampled with a parameters-free operator. Usually Nearest Neighbor or Bilinear upsampling are employed. When upsampling a low-resolution map, pixels close to object boundaries are often assigned to the wrong class, see Figure 3 (a). The idea behind GUM is to guide the upsampling operator through a guidance table of offsets vectors that steer sampling towards the correct semantic class. Figure 3 (b) depicts the Guided Upsampling Module. A Guidance Module predicts a high-resolution Guidance Offset Table. Then GUM performs a Nearest Neighbor upsampling by exploiting the Offset Table as steering guide. Each bidimensional coordinates vector of the regular sampling grid is summed with its corresponding bidimensional vector from the Guidance Offset Table. In Figure 3 the GUM module is presented in conjunction with Nearest Neighbor for simplicity, however, with simple modifications, GUM can be employed along with Bilinear operator.

Nearest Neighbor and Bilinear operators perform upsampling by superimposing a regular grid on the input feature map. Given G_i the regular input sampling grid, the output grid is produced by a linear transformation $\mathcal{T}_\theta(G_i)$. For the specific case of upsampling, $\mathcal{T}_\theta$ is simply defined as:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \begin{bmatrix} \theta & 0 \\ 0 & \theta \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \end{pmatrix}, \quad \text{with} \quad \theta \geq 1, \tag{1}$$

where $(x_i^s, y_i^s) \in G_i$ are source coordinates, (x_i^t, y_i^t) are target coordinates and θ represents the upsampling factor. Given V_i the output feature map and U_{nm} the input feature map, GUM can be defined as follows:

$$V_i = \sum_n^H \sum_m^W U_{nm} \delta(\lfloor x_i^s + p_i + 0.5 \rfloor - m) \delta(\lfloor y_i^s + q_i + 0.5 \rfloor - n) \tag{2}$$

where $\lfloor x_i^s + 0.5 \rfloor$ rounds coordinates to the nearest integer location and δ is a Kronecker delta function. Equation 2 represents a sum over the whole sampling grid U_{nm} where, through the Kronecker function, only a single specific location is selected and copied to the output. p_i and q_i represents the two offsets that shifts the sampling coordinates of each grid element in x and y dimensions respectively. They are the output of a function ϕ_i of i, the Guidance Module, defined as:

$$\phi_i = \begin{pmatrix} p_i \\ q_i \end{pmatrix} \tag{3}$$

Notice that V_i and U_{nm} are defined as bi-dimensional feature maps. The upsampling transformation is supposed to be consistent between channels therefore, equations presented in this section, generalize to multiple channels feature maps. In a similar way the bilinear sampling operator can be defined as:

$$V_i = \sum_n^H \sum_m^W U_{nm} \max(0, 1 - |x_i^s + p_i - m|) \max(0, 1 - |y_i^s + q_i - n|) \tag{4}$$

The resulting operator is differentiable with respect to U and p_i. We do not need the operator to be differentiable with respect to x_i^s because G_i is a fixed regular grid. Equations above follows the notation used by Jaderberg *et al.* in [10]. In the following paragraph we will briefly outline the connection between Guided Upsampling Module and Spatial Transformer Networks.

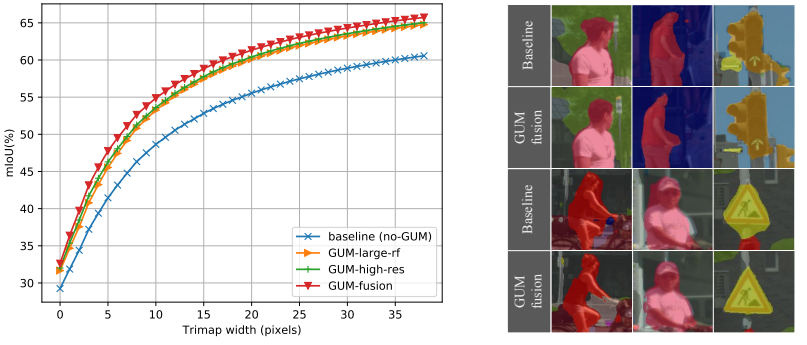
Connection with Spatial Transformer Networks (STN) [10] They introduce the ability for Convolutional Neural Networks to spatially warp the input signal with a learnable transformation. Authors of [10] separate an STN into three distinct modules: Localization Net, Grid Generator and Grid Sampler. Localization Net can be any function that outputs the transformation parameters conditioned on a particular input. Grid Generator takes as input the transformation parameters and warp a regular grid to match that specific transformation. Finally the Grid Sampler samples the input signal accordingly. Our Guided Upsampling Module can be interpreted as a Spatial Transformer Network where the Guidance Module plays the role of Localization Net and Grid Generator together. An STN explicitly outputs the parameters of a defined *a priori* transformation and then applies them to warp the regular sampling grid. GUM directly outputs offsets on x and y directions to warp the regular sampling grid without explicitly model the transformation. Grid Sampler plays the exact same role both in GUM and STN. Since Grid Sampler module is already implemented in major Deep Learning Frameworks *e.g.* PyTorch, TensorFlow, Caffe etc., integration of GUM within existing CNN architectures is quite straightforward.

Guidance module. The role of Guidance Module is to predict the Guidance Offset Table: the bidimensional grid that guides the upsampling process. The Guidance Module is a function which output is a tensor with specific dimensions: $HxWxC$ where H and W represents width and height of the high-resolution output semantic map and $C = 2$ is the dimension containing the two offset coordinates w.r.t x and y. We implemented the Guidance Module as a branch of our Neural Network, thus parameters are trainable end-to-end by backpropagation together with the whole network. We experimented three different designs for our Guidance Module and we named them *large-rf*, *high-res* and *fusion*.

- *large-rf* it is composed of three upsampling layers interleaved by Conv-BatchNorm-Relu blocks. It takes the output of the fusion module and gradually upsample it. This design relies on deep network layers activations with large receptive fields but doesn't

Guidance Module	baseline	large-rf	high-res	**fusion**
Large receptive-fields		✓		✓
High resolution details			✓	✓
mIoU (%)	67.5	69.36	69.29	**69.64**
FPS	37.3	26.3	34.8	**33.3**

Table 4: mIoU and FPS with different Guidance Modules. *large-rf* exploits signal from deeper layers. *high-res* exploits signal from early layers. *fusion* uses both. In bold the configuration adopted in the final model.



(a) (b)

Figure 4: (a) Trimap experiment for different Guidance Modules. Major improvement is w.r.t the baseline. (b) Examples of improved boundaries with GUM.

exploit high-resolution information. It is the most computationally demanding, due to the number of layers required.

- *high-res* it is composed by a single convolutional layer that takes as input a high-resolution activation map from the last Convolution before downsampling in the medium-resolution branch (see Section 3). The convolutional layer is a 1x1 kernel and maps the 32-dimensional feature space to a 2-dimensional feature space. It is almost free in terms of computational costs because, with our architecture, it only requires 64 additional parameters and the same number of additional per-pixel operations.
- *fusion* it lies in the middle between *large-rf* and *high-res* modules. It merges information coming from high-resolution and large-receptive-field activation maps using the *base sum* fusion module described in Section 3. It is a good compromise in terms of efficiency since it requires only two Conv-BatchNorm blocks and a single upsampling layer. Despite that, it is the one with most impact on performance because it exploit the required semantic information being at the same time faster than the iterative upsampling of *large-rf* design.

Table 4 reports mIoU on Cityscapes validation set and speed of the overall network in FPS. Best performance are achieved with *fusion* Guidance Module.

5 Boundaries analysis

To asses the behavior of our Guided Upsampling Network near object boundaries we performed a trimap experiment inspired by [2, 5, 11, 12]. The trimap experiment in [2, 5] was

run on Pascal VOC where semantic annotations include a specific class to be ignored in train and evaluation in correspondence with object boundaries. The trimap experiment was carried out by gradually increasing annotation borders with a morphological structuring element and considering for the evaluation only pixels belonging to the expanded boundaries. To the best of our knowledge we are the first to perform the trimap experiment on Cityscapes dataset. Since there is no boundary class to expand we decided to implement the experiment in a different but equivalent way: for each object class independently we computed the distance transform on ground-truth maps. Then we performed the trimap experiment by gradually increasing the threshold on our computed distance transform map to include pixels at different distances from object boundaries. Figure 4 shows a qualitative and quantitative comparison of the three Guidance Modules *i.e*. *large-rf*, *high-res* and *fusion* with respect to the *baseline*, where baseline is the exact same network with bilinear upsampling instead of GUM. There is a clear advantage in using GUM versus the baseline. The type of Guidance Module does not drastically affect the results even though GUM with *fusion* achieve slightly higher mIoU levels.

6 Comparison with the state-of-the-art

In Table 5 we reported performance of Guided Upsampling Network along with state-of-the-art methods on Cityscapes test set. Segmentation quality has been evaluated by Cityscapes evaluation server and it is reported in the official leaderboard[1]. FPS in Table 5 have been estimated on a single Titan Xp GPU. For fairness we only included algorithms that declare their running time on Cityscapes leaderboard, even though DeepLabv3+[5] has been listed in Table 5 as a reference for accuracy-oriented methods. Usually, methods that do not care about processing time, are computationally heavy. Most of them *e.g*. PSPNet, DeepLabv3 [5, 25] achieve very high mIoU levels, *i.e*. DeepLabv3+ is the best published model to date, reaching 81.2%, but they adopt very time-consuming multi-scale testing to increase accuracy. Our Guided Upsampling Network achieve 70.4% of mIoU on Cityscapes test set without any postprocessing. To the best of our knowledge this is the highest mIoU for a published method running at >30 FPS. It performs even better than some methods like Adelaide, Dilation10 etc. that do not care about speed.

7 Conclusions

We proposed a novel network architecture to perform real-time semantic segmentation of street scene images. It consists of a multiresolution architecture to jointly exploit high-resolution textures and large context information. We introduced a new module named Guided Upsampling Module to improve upsampling operators by learning a transformation conditioned on high-resolution details. We included GUM in our network architecture and we experimentally demonstrated performance improvements with low additional comptutational costs. We evaluated our network on the Cityscapes test dataset showing that it is able to achieve 70.4% mIoU while running at 33.3 FPS on a single Titan Xp GPU. Further details and a demo video can be found in our project page: http://www.ivl.disco.unimib.it/activities/semantic-segmentation.

[1] https://www.cityscapes-dataset.com/benchmarks/

Name	Subsampling	mIoU(%)	FPS
SegNet [1]	4	57.0	26.4
ENet [16]	2	58.3	121.5
SQ [20]	no	59.8	26.4
CRF-RNN [26]	2	62.5	2.2
DeepLab [3]	2	63.1	0.4
FCN-8S [15]	no	65.3	4.9
Adelaide [24]	no	66.4	0.05
Dilation10 [25]	no	67.1	0.4
ICNet [22]	no	69.5	47.9
ERFNet [19]	2	69.7	52.6
GUN (ours)	**2**	**70.4**	**33.3**
DeepLabv3+[5]	no	81.2	n\a

Table 5: Comparison with state-of-the-art methods on Cityscapes test set sorted by increasing mIoU. Our method in boldface.

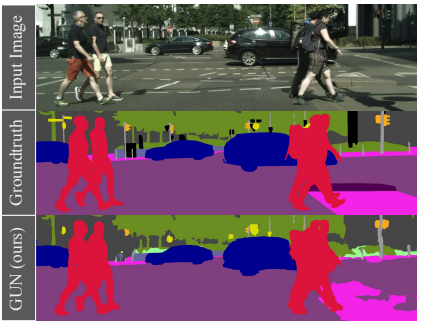


Figure 5: From top to bottom respectively input image, ground-truth and prediction obtained with our Guided Upsampling Net.

8 Acknowledgements

The research leading to these results has received funding from TEINVEIN: TEcnologie INnovative per i VEicoli Intelligenti, Unique Project Code: E96D17000110009 - Call "Accordi per la Ricerca e l'Innovazione", cofunded by POR FESR 2014-2020 Regional Operational Programme, European Regional Development Fund.
We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan Xp GPU used for this research.

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] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[4] 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, 2018.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[6] 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. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[9] 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.

[10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[11] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[12] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

[13] 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.

[14] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016.

[15] 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.

[16] 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.

[17] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *31st International Conference on Machine Learning (ICML)*, number EPFL-CONF-199822, 2014.

[18] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. *arXiv preprint*, 2017.

[19] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2018.

[20] Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, et al. Speeding up semantic segmentation for autonomous driving. In *MLITS, NIPS Workshop*, 2016.

[21] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016.

[22] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[23] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. *arXiv preprint arXiv:1704.08545*, 2017.

[25] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[26] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.