HyperSeg: Patch-wise Hypernetwork for Real-time Semantic Segmentation — Supplemental material —

Yuval Nirkin*
Facebook AI & Bar-Ilan University

Lior Wolf
Facebook AI & Tel Aviv University

Tal Hassner
Facebook AI

A. Feature division algorithm

Algorithm. 1 describes the signal channel division algorithm referred to in Sec. 3.2. The algorithm starts by dividing the channels, C, into units of size s_u. In our experiments $s_u = max(g_{w_0}, \ldots, g_{w_n})$, which assures that the divided channels will be divisible by their corresponding g_{w_i}, which are all power of 2.

Each weight is first allocated a single unit in order to make sure minimal channel allocation for each of the weights. The units are divided by their total number relative to the sum of the weights, starting from the large weights. This gives priority to the smaller weights, which receive the remainder of the allocations that were rounded down.

B. Model details

The hyperparameters of each of our five models are laid out in Tab. S1. Variables r_i are the reduction factors, corresponding to each of the features maps, F_i, from b, defined in Sec. 3.1. g_{w_i}, also defined in Sec. 3.1, equal 16 across all levels for our earliest model, HyperSeg-L (PASCAL VOC), and their values were experimentally adapted for the newer models trained on CamVid and Cityscapes.

The last rows detail the number of feature map channels in each m_i. A single arrow, $C_{in} \rightarrow C_{out}$, denotes a single 1×1 convolution $pw_1 : \mathbb{R}^{C_{in}\times\frac{H}{2^i}\times\frac{W}{2^i}} \rightarrow \mathbb{R}^{C_{out}\times\frac{H}{2^i}\times\frac{W}{2^i}}$, and two arrows, $C_{in} \rightarrow C_{hidden} \rightarrow C_{out}$, specify the channels of the full meta block (described in Sec. 3.2):

$$pw_1 : \mathbb{R}^{C_{in}\times\frac{H}{2^i}\times\frac{W}{2^i}} \rightarrow \mathbb{R}^{C_{hidden}\times\frac{H}{2^i}\times\frac{W}{2^i}}, \quad (1)$$

$$dw : \mathbb{R}^{C_{hidden}\times\frac{H}{2^i}\times\frac{W}{2^i}} \rightarrow \mathbb{R}^{C_{hidden}\times\frac{H}{2^i}\times\frac{W}{2^i}}, \quad (2)$$

$$pw_2 : \mathbb{R}^{C_{hidden}\times\frac{H}{2^i}\times\frac{W}{2^i}} \rightarrow \mathbb{R}^{C_{out}\times\frac{H}{2^i}\times\frac{W}{2^i}} \quad (3)$$

Our PASCAL VOC model was trained using the cross entropy loss; all other models were trained using bootstrapped cross entropy loss [8].

*Performed this work while an intern at Facebook.

C. Additional ablation studies

Ablation study on PASCAL VOC. We tested multiple variants of our method, to show the effects of employing spatially varying convolutions and to evaluate the contribution of the positional encoding. We describe these variants using the following terminology: 1×1 denotes evaluating the entire image as a single patch. That is, we only generate a single set of weights per input image. For this variant, we completely remove network h. 16×16 is the original number of patches used for reporting our results on PASCAL VOC in Tab. 2.

Our ablation results are reported in Tab. S2. Evidently, the larger the grid size, the better our accuracy. The contribution of the positional encoding is significant, given that the gap between the two best previous methods is 0.1%, as can be seen in Tab. 2. As evident from the reported FPS column, the 1×1 variant is the fastest because it does not require unoptimized operations used for the DPWConv and because the network h is absent.

Ablation study on Cityscapes. We test the effect of varying $g_{w_i}, i \in [1, 5]$ in our HyperSeg-M model on resolutions of 1536×768 and report results in Tab. S3. We fix the ratio between the groups relative to $|\theta^{m_i}|$ while maintaining multiples of 2. We start by setting $g_{w_1} = 1$ for the test in the first row and then double the group number in each subsequent test. The number of parameters and flops of w_i decreases as the number of groups increases, according to Eq. 6 and Eq. 7. Surprisingly, the experiment in the middle row provides the best GFLOPs / accuracy trade-off, achieving the best accuracy with fewer GFLOPs and parameters than the first two tests.

D. Open source repositories

The open source repositories used in Tables 2, 3, and 4, to report information about previous methods, which was not available otherwise, are listed in Tab. S4. The protocol for computing the FPS, GFLOPs, and trainable parameters, is described in Sec. 4.

Algorithm 1 Divides the channels, C, in unit size, s_u, into chunks relative to the weights, $w_0, \ldots, w_n$.

procedure DIVIDE_CHANNELS($C, s_u, w_0, \ldots, w_n$)
  $total\_units \leftarrow \frac{C}{s_u}$
  $w \leftarrow sort(w_0, \ldots, w_n)$ ▷ Descending order
  $r \leftarrow \frac{total\_units}{\sum_{i=0}^{n} w_i}$ ▷ Units to weights ratio
  $out \leftarrow \{s_u |$ for each $w_i \in w\}$ ▷ Each weight group should be allocated with at least one unit
  $total\_units \leftarrow total\_units - |out|$
  $i \leftarrow 0$
  while $total\_units \neq 0$ **do**
    if $i = n$ **or** $\lfloor w_i \cdot r \rfloor \leq 1$ **then**
      $curr\_units \leftarrow total\_units$
    else
      $curr\_units \leftarrow \lfloor w_i \cdot r \rfloor - 1$
    $out_i \leftarrow out_i + curr\_units \cdot s_u$
    $total\_units \leftarrow total\_units - curr\_units$
    $i \leftarrow i + 1$
  return out

Params	HyperSeg-L (PASCAL VOC)	HyperSeg-S (Cityscapes)	HyperSeg-M (Cityscapes)	HyperSeg-S (CamVid)	HyperSeg-L (CamVid)
Backbone	EfficientNet-B3	EfficientNet-B1	EfficientNet-B1	EfficientNet-B1	EfficientNet-B1
Resolution	512×512	1536×768	1024×512	768×576	1024×768
$r_1, \ldots, r_5$	1/4, 1/4, 1/4, 1/4, 1/4	-, 2/5, 1/4, 1/5, 1/6	1/4, 1/4, 1/4, 1/4, 1/4	1/4, 1/4, 1/4, 1/4, 1/4	1/4, 1/4, 1/4, 1/4, 1/4
$g_{w_0}, \ldots, g_{w_5}$	16, 16, 16, 16, 16, 16	-, 4, 16, 8, 16, 32	-, 4, 16, 8, 16, 32	-, 8, 16, 32, 32, 64	8, 8, 16, 32, 32, 64
m_5 channels	$98 \rightarrow 96$	$130 \rightarrow 32$	$82 \rightarrow 64$	$82 \rightarrow 64$	$82 \rightarrow 64$
m_4 channels	$132 \rightarrow 34$	$62 \rightarrow 16$	$94 \rightarrow 32$	$94 \rightarrow 32$	$94 \rightarrow 32$
m_3 channels	$48 \rightarrow 96 \rightarrow 12$	$26 \rightarrow 8$	$44 \rightarrow 16$	$44 \rightarrow 16$	$44 \rightarrow 16$
m_2 channels	$22 \rightarrow 44 \rightarrow 8$	$14 \rightarrow 28 \rightarrow 8$	$24 \rightarrow 48 \rightarrow 16$	$24 \rightarrow 48 \rightarrow 16$	$24 \rightarrow 48 \rightarrow 16$
m_1 channels	$16 \rightarrow 32 \rightarrow 6$	$26 \rightarrow 52 \rightarrow 19$	$34 \rightarrow 68 \rightarrow 19$	$22 \rightarrow 44 \rightarrow 12$	$22 \rightarrow 44 \rightarrow 16$
m_0 channels	$11 \rightarrow 22 \rightarrow 21$	-	-	-	$21 \rightarrow 42 \rightarrow 12$

Table S1: *Model details.* Each row represents a different model hyperparameter and each column a different model, where "HyperSeg-<size> (dataset)" is the model's template name (see Sec. 4) and the dataset on which it was trained on. "-" denotes that the decoder level corresponding to the specific hyperparameter was omitted.

Grid size	Positional encoding	mIoU (%)	FPS
1×1	✗	77.56	**46.8**
4×4	✗	78.92	22.4
8×8	✗	80.23	26.9
16×16	✗	80.33	28.2
16×16	✓	**80.61**	26.8

Table S2: *Ablation study on PASCAL VOC 2012, val. set [4].* Each row represents a different model, trained with the specified grid size, with or without positional encoding.

Groups	mIoU (%)	GFLOPs	$\lvert\theta^w\rvert$ (M)	Params (M)
1, 2, 4, 4, 8	77.1	18.0	1.2	11.1
2, 4, 8, 8, 16	77.5	17.3	0.6	10.4
4, 8, 16, 16, 32	**78.0**	16.9	0.3	10.1
8, 16, 32, 32, 64	77.8	16.7	0.1	10.0
16, 32, 64, 64, 128	77.2	**16.6**	**0.1**	**9.9**

Table S3: *Ablation study on Cityscapes, val. set [3].* *The* Groups *column represents:* $g_{w_1}, \ldots, g_{w_n}$.

E. Convolution and batch normalization fusion

Following others [7], we fuse the convolution and batch normalization operations in the inference stage for improving the runtime performance. For regular convolutions, the batch normalization operation is fused with its prior convolution. The batch normalization operations following DPWConv are fused with the corresponding weight mapping layer. We next provide more details on these steps.

Method	URL
Auto-DeepLab-L [5]	https://github.com/MenghaoGuo/AutoDeeplab
DeepLabV3 [2]	https://github.com/pytorch/vision
ERFNet [9]	https://github.com/Eromera/erfnet_pytorch
ESPNetV2 [6]	https://github.com/sacmehta/ESPNetv2
SwiftNetRN-18 [7]	https://github.com/orsic/swiftnet
BiSeNetV1 [11]	https://github.com/CoinCheung/BiSeNet
BiSeNetV2 [10]	https://github.com/CoinCheung/BiSeNet

Table S4: *List of open source repositories used for comparing to previous methods.*

The batch normalization operation can be described in the form of matrix multiplication, $\theta^{BN} \cdot x + b^{BN}$, for which $\theta^{BN}_{i,i} = \frac{\gamma_i}{\sqrt{\sigma_i^2+\epsilon}}$ on the diagonal and zero everywhere else, and $b^{BN}_i = \beta_i - \gamma_i \frac{\mu_i}{\sqrt{\sigma_i^2+\epsilon}}$ for $i \in [0, C)$, where C is the number of feature channels, γ is the scaling factor, β is the shifting factor, μ and σ are the mean and standard deviation, respectively, and ϵ is a scalar constant used for numeric stabilization.

The convolution operation can also be written as a matrix multiplication by reshaping its weights, θ^*, and input feature map, x, such that $\theta^* \in \mathbb{R}^{C \times C_{in} \times k^2}$ and $\tilde{x} \in \mathbb{R}^{C_{in} \times k^2}$, where $\tilde{x}_{i,j}$ is the $k \times k$ neighborhood in location (i, j) of the original feature, x. The combined convolution and batch normalization operations can then be written as:

$$O_{i,j} = \theta^{BN} \cdot (\theta^* \cdot \tilde{x}_{i,j} + b^*) + b^{BN}. \quad (4)$$

The fused convolution operation will then have the weights $\tilde{\theta}^* = \theta^{BN} \cdot \theta^*$ and bias $\tilde{b}^* = \theta^{BN} \cdot b^* + b_{BN}$.

Similarly, the DPWConv operation on a patch location (m, n) followed by batch normalization is:

$$O_{i,j} = \theta^{BN} \cdot [(\theta^w \cdot \phi_{m,n}) \cdot \tilde{x}_{i,j} + b^w] + b^{BN}, \quad (5)$$

where θ^w are the weights of the weight mapping layer and $\phi_{m,n}$ is the signal corresponding to the patch in location (m, n). The batch normalization operation can then be fused into the weight mapping layer using the adjusted weights, $\tilde{\theta}^w = \theta^{BN} \cdot \theta^w$, and bias, $\tilde{b}^w = \theta^{BN} \cdot b^w + b^{BN}$.

F. Additional qualitative results

Fig. S1 provides qualitative results on PASCAL VOC 2012 val. set [4]. In the first four rows we have specifically chosen samples with different classes to best demonstrate the performance of our model. The last two rows offer failure cases, top left: boat classified as a chair; bottom left: the model failed to detect the bottles from a top view; top right: dog classified as a cat; and bottom right: sheep classified as a dog.

Fig. S2 shows qualitative results of our HyperSeg-L model on the CamVid dataset test set [1]. The first four rows display predictions on different scenes and the last two rows demonstrate failures of our model: in the first row a bicyclist is partly segmented as a pedestrian, and in the last row our model fails to detect a sign.

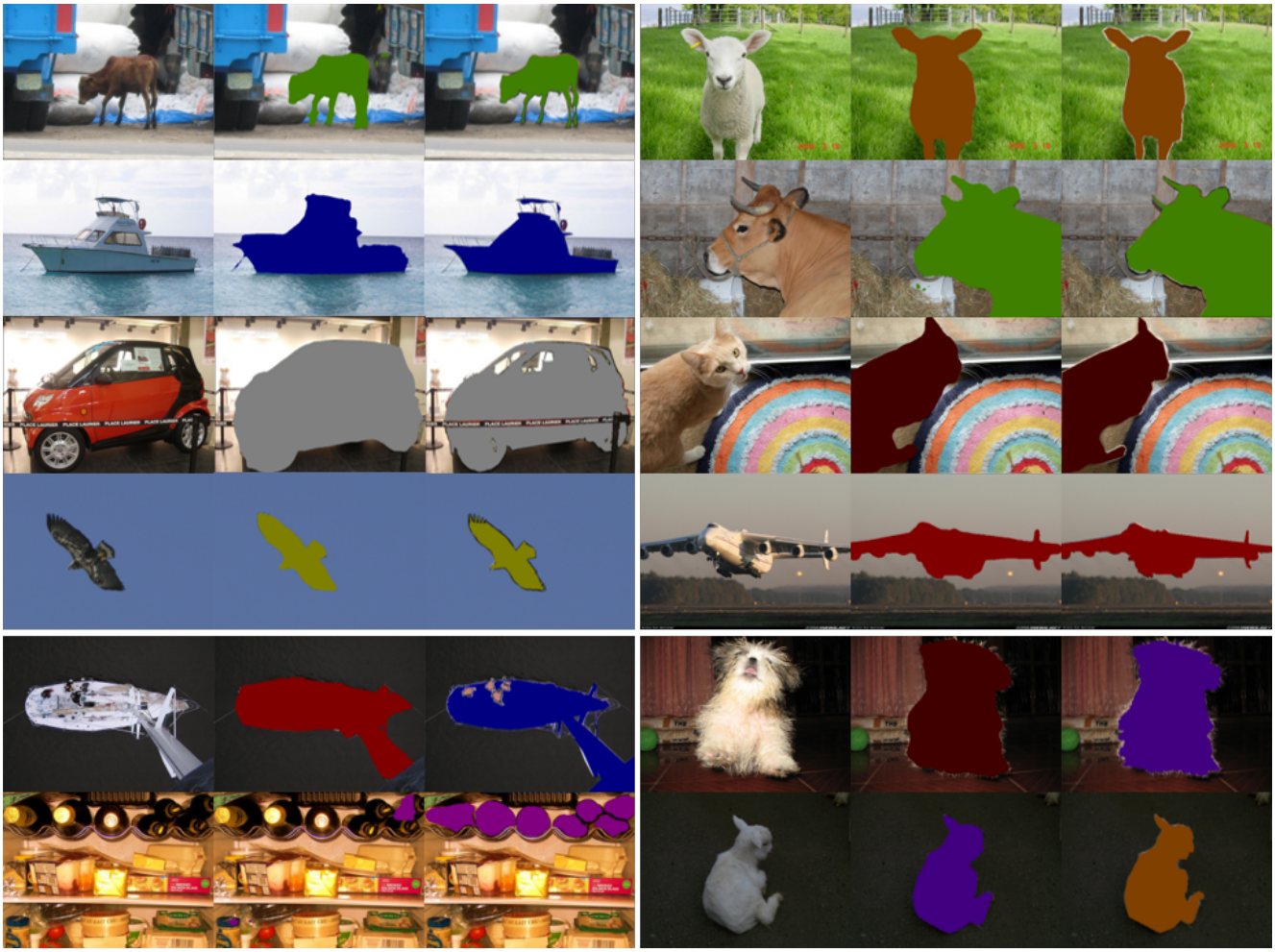

Figure S1: *Qualitative results on PASCAL VOC 2012 validation set [4].* First and 4th columns: input image, 2nd and 5th columns: our predictions, and 3rd and final column: ground truth. The first four rows demonstrate how our model performs on different classes. The last two rows present failure cases of our model.

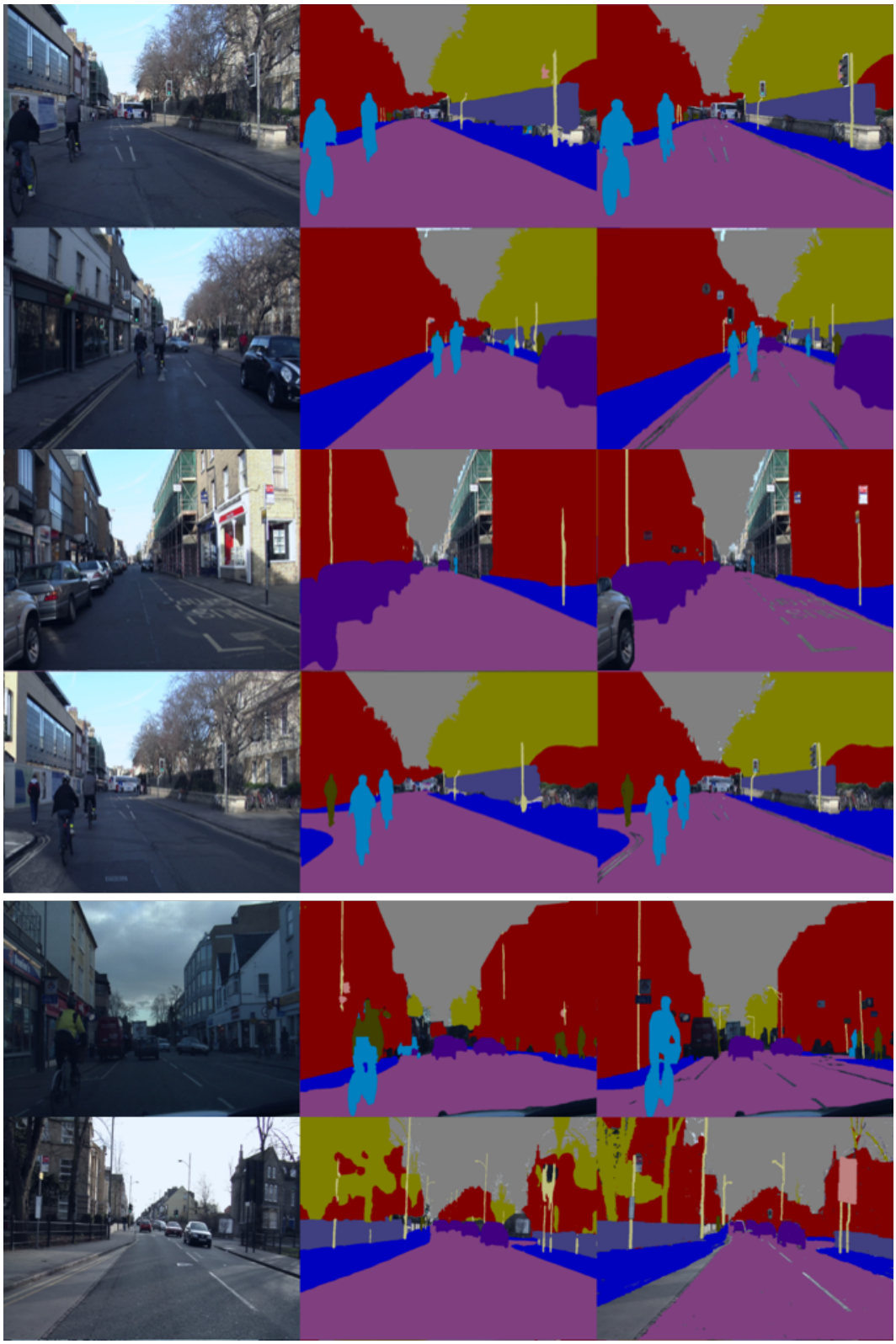

Figure S2: *Qualitative results on CamVid test set [1].* The columns represent: input (left), prediction (center), and ground truth (right). The first four rows provide samples from different scenes, and the last two rows demonstrate failure cases.

References

[1] 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, 5

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

[3] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3213–3223, 2016. 2

[4] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. 2, 3, 4

[5] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 82–92, 2019. 3

[6] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Proceedings of the european conference on computer vision (ECCV)*, pages 552–568, 2018. 3

[7] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12607–12616, 2019. 2, 3

[8] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014. 1

[9] 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, 2017. 3

[10] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *arXiv preprint arXiv:2004.02147*, 2020. 3

[11] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Eur. Conf. Comput. Vis.*, pages 325–341, 2018. 3