Supplementary Material: Content-aware Token Sharing for Efficient Semantic Segmentation with Vision Transformers

Chenyang Lu*, Daan de Geus*, Gijs Dubbelman
Eindhoven University of Technology
{c.lu.2, d.c.d.geus, g.dubbelman}@tue.nl

In this document, we provide the following material in addition to the main manuscript:

- More extensive implementation details (Section 1).
- Detailed results for the dynamic token sharing experiment (Section 2).
- Additional qualitative results, showing predictions by both the policy network and the complete semantic segmentation network (Section 3).

The code for Content-aware Token Sharing (CTS) is available through https://tue-mps.github.io/CTS.

1. Implementation details

Below, we provide the implementation details for the networks used in our experiments. Note that, for our main experiments, we conducted each experiment 5 times, and report the median mIoU.

1.1. Policy network

Our default Content-aware Token Sharing (CTS) policy network, used to identify what superpatches should share a token, is based on EfficientNet [10]. Specifically, we use EfficientNet-Lite0, a light version of EfficientNet-B0, pre-trained on ImageNet-1K [7], using the code and pre-trained model weights from the *timm* repository [12]. To be able to make a prediction per superpatch, we first output the features before the final classification head. By design of the network, these features have resolution $H_p \times W_p$ that is 32 times smaller than the resolution of the input images. As all segmentation networks used in this work use a patch size of 16×16 pixels, a superpatch is 32×32 pixels. Therefore, each of the $H_p \cdot W_p$ features represent one superpatch. To get a policy prediction per superpatch, we add one final 1×1 convolutional layer that maps the features to two classes, *i.e.*, (1) the superpatch contains a single class and (2) the superpatch contains multiple classes. We train the policy network separately for ADE20K [14], Pascal Context [6] and Cityscapes [3]. The policy network is always trained for 50k iterations with batches of 8 image crops, a learning rate of 5×10^{-5} and a weight decay of 10^{-3}, using the AdamW optimizer [5].

Large and small policy networks. In Table 2 of the main manuscript, we also provide results for a large and small policy network. The large policy network is EfficientNet-B1 [10], pre-trained on ImageNet-1K [7], using the code and pre-trained weights from the *timm* repository [12]. All hyperparameters are the same as for the default policy network. The small policy network is a custom neural network that has seven 3×3 convolutional layers, with 128 output channels per convolution, and a final 1×1 convolution to output a prediction for two classes. Strided convolutions are used to reduce the resolution by a factor of 32. This network is not pre-trained, and we train it with a learning rate of 10^{-3}.

1.2. Segmenter

In our main experiments, we apply our CTS to a transformer-based semantic segmentation network, Segmenter [9]. For these experiments, we implement CTS in the official, publicly available code of Segmenter, and we use the original training settings. That means that we use a batch size of 8 for all experiments, use the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, a weight decay of 0.0, and a polynomial decay learning rate schedule. For ADE20K, we train for 64 epochs with an initial learning rate of 0.001, resize the shortest side of images to 512 pixels, and take random crops of 512×512 pixels. For the *ViT-L (640)* setting in Figure 5 of the main manuscript, we resize the shortest side of images to 640 pixels, and take random crops of 640×640 pixels. For Pascal Context, we train for 256 epochs with an initial learning rate of 0.001, resize the shortest side of images to 520 pixels,

*Both authors contributed equally.

and take random crops of 480×480 pixels. For Cityscapes, we train for 216 epochs with an initial learning rate of 0.01, and take random crops of 768×768 pixels.

DeiT and BEiT. For our experiment with different backbones, we use DeiT-B and BEiT-B in addition to the standard ViT backbones. For DeiT-B [11], we follow the settings from Segmenter, and initialize the backbone with distilled weights from the *timm* repository [12], pre-trained on ImageNet-1K. In the experiments with the BEiT-B backbone [1], we also use the code and pre-trained weights from *timm*, but we make minor changes for compatibility: (1) We use the AdamW optimizer with an initial learning rate of 10^{-5} and weight decay of 10^{-3} because the SGD optimizer made training unstable. (2) Because BEiT works with relative position embeddings between all tokens, these have to be shared too, when a superpatch shares a token. We empirically find that picking the relative position to one of the patches in a superpatch works just as well as taking the mean of the relative positions to each of them, while being significantly faster.

1.3. UPerNet

ViT + UPerNet. For our experiments with the UPerNet [13] decoder in combination with the standard ViT backbone [4, 8], we use the publicly available code and hyperparameters from the *mmsegmentation* repository [2], and implement CTS in this code. This means that we train these networks for 160k iterations, with a batch size of 16. We first resize the shortest side of images to 512 pixels, and then take random crops of 512×512 pixels. We use the AdamW optimizer, with a learning rate of 6×10^{-5} and a weight decay of 0.01.

BEiT + UPerNet. When applying our CTS to state-of-the-art network BEiT-L + UPerNet [1, 13], we also use the publicly available code and hyperparameters from the *mmsegmentation* repository [2]. The weights of BEiT-L are initialized using the publicly available weights from *timm* [12]. We train the networks with a batch size of 8, for 320k iterations. We first resize the shortest side of images to 640 pixels, and then take random crops of 640×640 pixels. The networks are optimized using AdamW, with a learning rate of 2×10^{-5} and a weight decay of 0.05. Again, we handle the relative positions embeddings as described for BEiT-B in Section 1.2.

2. Dynamic token sharing

In Section 5.5 of the main manuscript, we analyze whether it is desirable to let the number of shared tokens, S, depend on the complexity of the image. To evaluate this, we feed each image through the policy network, and use its predictions to determine how many superpatches can share a token. If the predicted score for a superpatch is above a threshold τ, we assume that this superpatch contains a single class, and can therefore share a token. The resulting number of superpatches that can share a token for each image is then given by S^*. Then, we make sure that a maximum of S^* superpatches are shared per image. Specifically, from a set of models trained with different token sharing settings S, we pick the model with the highest token sharing setting S for which $\mathrm{S} < \mathrm{S}^*$. Subsequently, we let the S highest-scoring superpatches share a token, and feed the resulting tokens through the segmentation model trained with setting S. In Table 1, we provide the results for this analysis for different values of threshold τ. It can be observed that, for all evaluated thresholds, the combined mIoU is consistently higher than the highest mIoU among all the individual models. Note that this high performance is obtained by processing more images by the models with higher token reduction settings, which have a lower individual mIoU when processing all images. Moreover, the average token reduction is above the 30% reduction setting which was found to be optimal when the number of shared tokens is fixed. This makes dynamic token sharing for semantic segmentation an interesting topic for future research.

3. Qualitative results

Policy network In Figure 1 and Figure 2, we show qualitative examples of predictions by our CTS policy network. In the figures we can see that, for all three datasets, the policy network learns to predict what superpatches contain a single semantic class quite accurately. As a result, the S highest-scoring superpatches, *i.e.*, the superpatches that share a token, mostly consist of a single semantic class, as is the intended behavior.

Semantic segmentation predictions In Figure 3 and Figure 4, we show qualitative examples of semantic segmentation predictions by Segmenter [9] with our CTS. These figures show that the network mostly predicts a single class for the superpatches that share a token, which is intended behavior. Additionally, we do not observe any qualitative artifacts in the predictions that result from sharing tokens.

Finally, we show the results of our CTS with state-of-the-art semantic segmentation network BEiT-L + UPerNet [1, 13] in Figure 5. These results show that it is possible to achieve state-of-the-art segmentation quality with our CTS, while also achieving a throughput increase of over 60%.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers. In *ICLR*, 2022. 2, 8

[2] MMSegmentation Contributors. MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 2

Token reduction	mIoU (indiv. models)	Processed images with different thresholds τ						
		$\tau = 0.35$	$\tau = 0.4$	$\tau = 0.45$	$\tau = 0.5$	$\tau = 0.55$	$\tau = 0.6$	$\tau = 0.65$
0%	45.0	1	2	3	8	12	22	32
12%	44.9	77	87	105	134	163	196	232
23%	45.0	150	191	219	244	270	289	316
30%	45.1	366	398	436	455	460	465	463
38%	44.7	446	427	410	397	394	392	386
46%	44.4	960	895	827	762	701	636	571
Average token reduction		38.2%	37.4%	36.5%	35.5%	34.6%	33.6%	32.4%
Combined mIoU		45.4	**45.8**	45.5	45.3	45.3	45.2	45.4

Table 1. **Dynamic token sharing ablation.** Results for the dynamic token sharing experiment for different settings of threshold τ. Experiments with Segmenter [9] and ViT-S/16 [4] on the ADE20K validation set [14].

[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 *CVPR*, 2016. 1, 5, 7

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 2, 3

[5] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 1

[6] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*, 2014. 1, 5, 7

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015. 1

[8] Andreas Peter Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. *TMLR*, 2022. 2

[9] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. In *ICCV*, 2021. 1, 2, 3

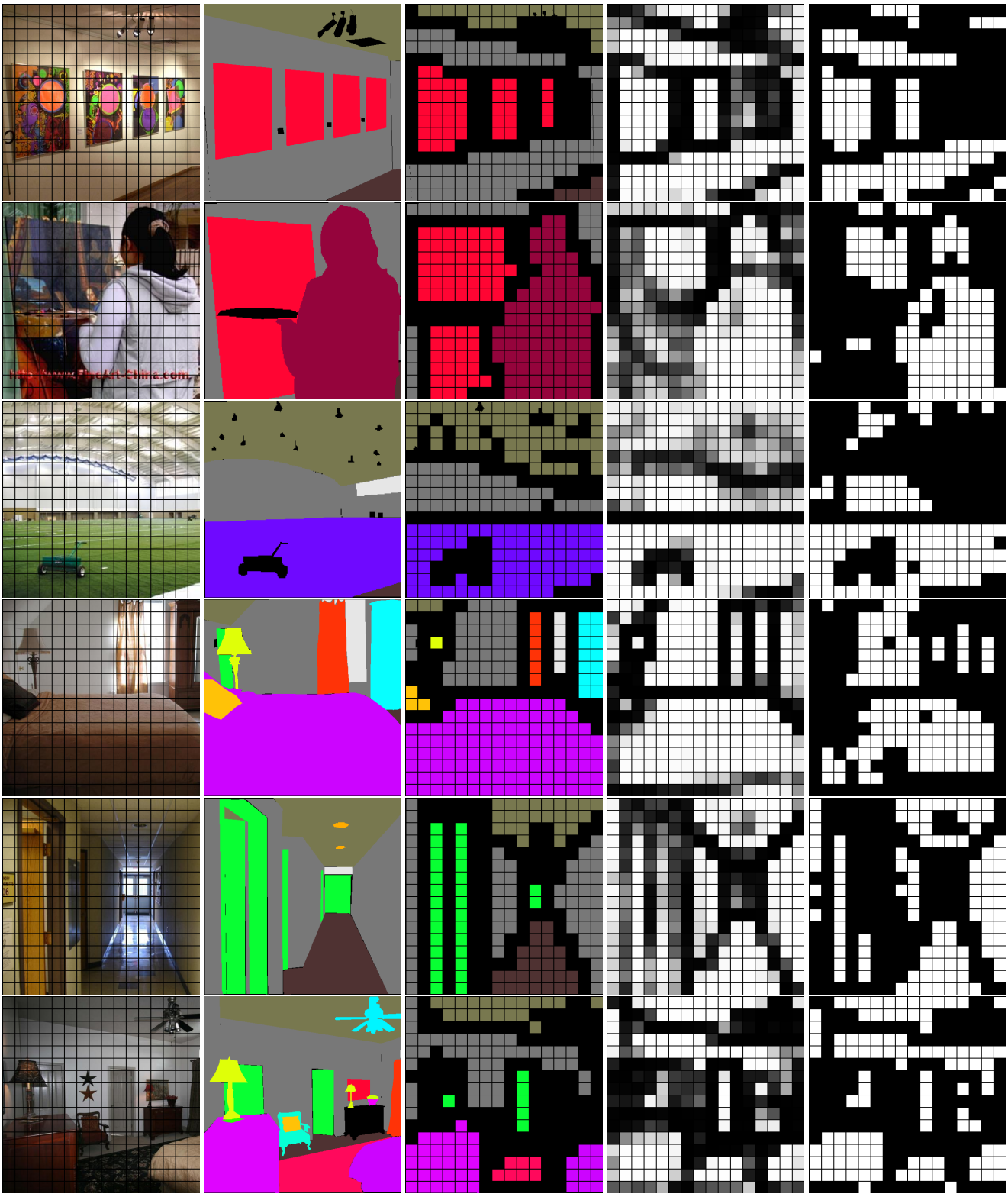
[10] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, 2019. 1

[11] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 2

[12] Ross Wightman. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models, 2019. 1, 2

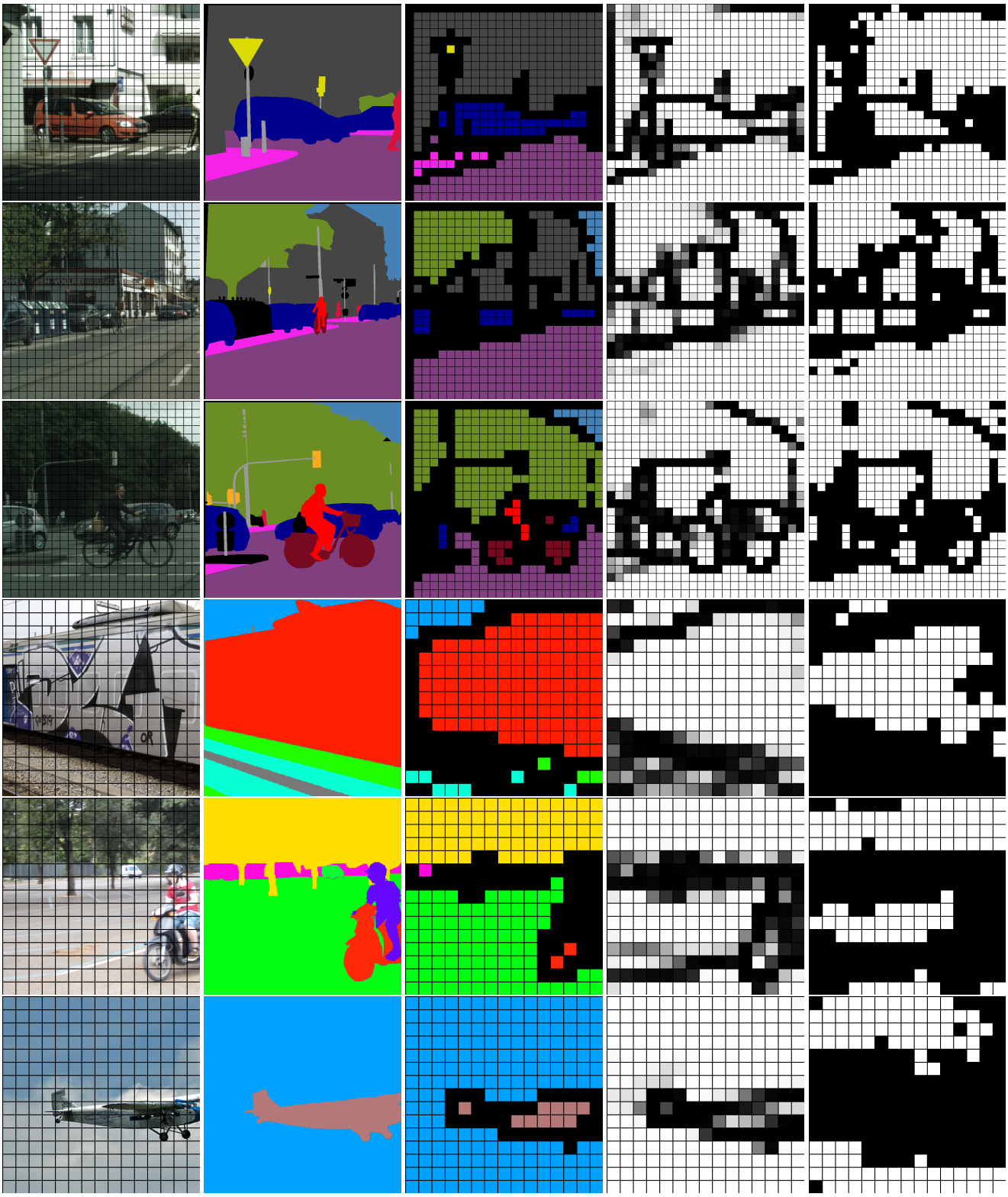
[13] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified Perceptual Parsing for Scene Understanding. In *ECCV*, 2018. 2, 8

[14] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene Parsing through ADE20K Dataset. In *CVPR*, 2017. 1, 3, 4, 6, 8



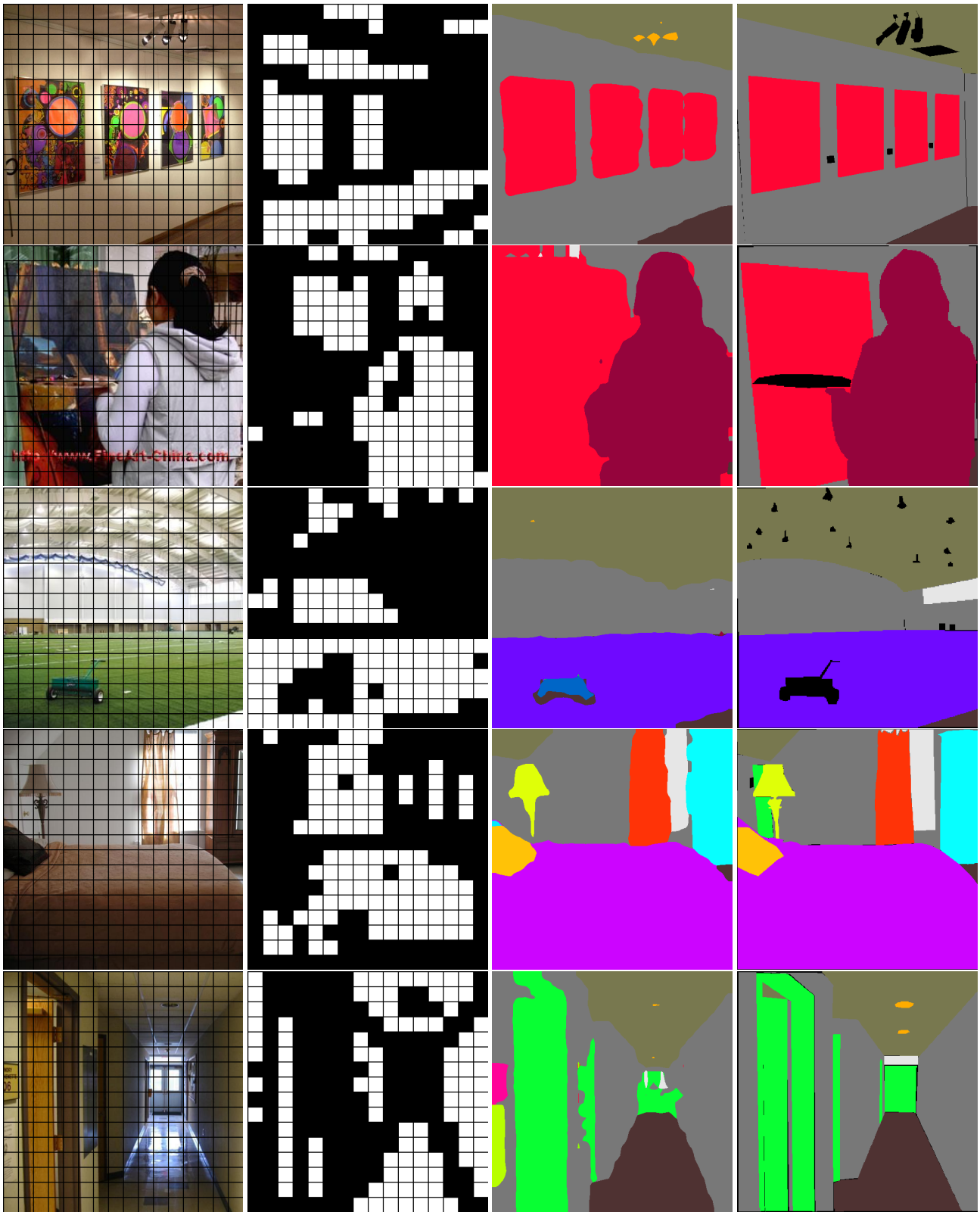
(a) Image with grid of super-patches (b) Semantic segmentation ground truth (c) Superpatches with a single class (d) Policy network prediction (e) S highest-scoring super-patches (*i.e.*, used policy)

Figure 1. **Qualitative results policy network.** Predictions by our content-aware token sharing policy network, on image crops from the ADE20K validation set [14].



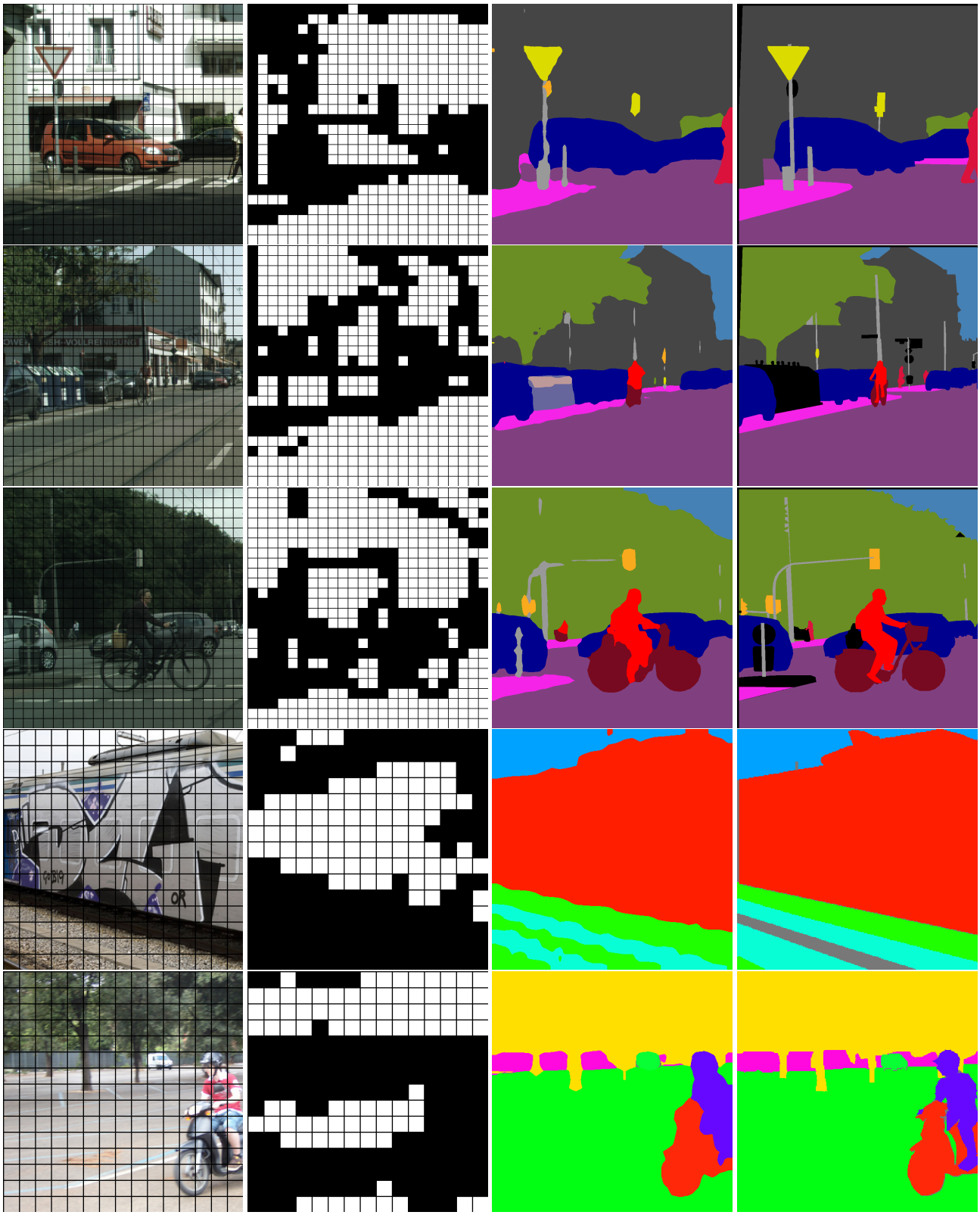
(a) Image with grid of superpatches (b) Semantic segmentation ground truth (c) Superpatches with a single class (d) Policy network prediction (e) S highest-scoring superpatches (*i.e.*, used policy)

Figure 2. **Qualitative results policy network.** Predictions by our content-aware token sharing policy network, on image crops from the Cityscapes val set (top three images) [3] and Pascal Context validation set (bottom three images) [6].



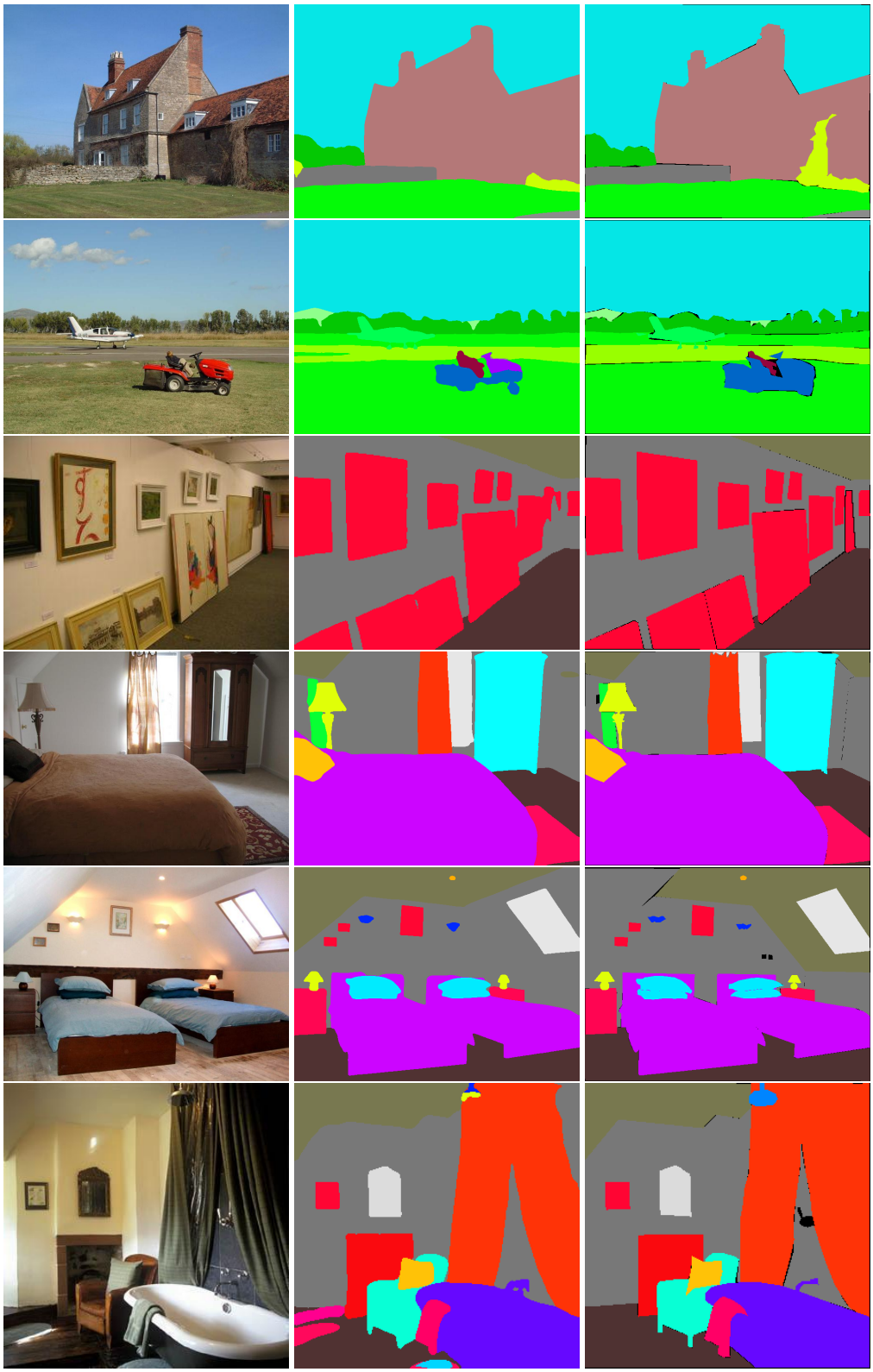
(a) Input image (b) Used token sharing policy (c) Segmentation prediction (d) Segmentation ground truth

Figure 3. **Qualitative results semantic segmentation from policy.** Predictions by Segmenter with ViT-S/16 and our CTS, on image crops from the ADE20K validation set [14].



(a) Input image (b) Used token sharing policy (c) Segmentation prediction (d) Segmentation ground truth

Figure 4. **Qualitative results semantic segmentation from policy.** Predictions by Segmenter with ViT-S/16 and our CTS, on image crops from the Cityscapes val set (top three images) [3] and Pascal Context validation set (bottom two images) [6].



(a) Input image (b) Segmentation prediction (c) Segmentation ground truth

Figure 5. **Qualitative results semantic segmentation state-of-the-art.** Examples of results by BEiT-L + UPerNet [1, 13], a state-of-the-art semantic segmentation approach, with our CTS, on the ADE20K validation set [14]. With our CTS, we are able to increase the throughput of this network by more than 64%, without decreasing the segmentation quality.