Segmenter: Transformer for Semantic Segmentation

Robin Strudel* Inria†
Ricardo Garcia* Inria†
Ivan Laptev Inria†
Cordelia Schmid Inria†

Abstract

Image segmentation is often ambiguous at the level of individual image patches and requires contextual information to reach label consensus. In this paper we introduce Segmenter, a transformer model for semantic segmentation. In contrast to convolution-based methods, our approach allows to model global context already at the first layer and throughout the network. We build on the recent Vision Transformer (ViT) and extend it to semantic segmentation. To do so, we rely on the output embeddings corresponding to image patches and obtain class labels from these embeddings with a point-wise linear decoder or a mask transformer decoder. We leverage models pre-trained for image classification and show that we can fine-tune them on moderate sized datasets available for semantic segmentation. The linear decoder allows to obtain excellent results already, but the performance can be further improved by a mask transformer generating class masks. We conduct an extensive ablation study to show the impact of the different parameters, in particular the performance is better for large models and small patch sizes. Segmenter attains excellent results for semantic segmentation. It outperforms the state of the art on both ADE20K and Pascal Context datasets and is competitive on Cityscapes.

1. Introduction

Semantic segmentation is a challenging computer vision problem with a wide range of applications including autonomous driving, robotics, augmented reality, image editing, medical imaging and many others [27, 28, 45]. The goal of semantic segmentation is to assign each image pixel to a category label corresponding to the underlying object and to provide high-level image representations for target tasks, e.g. detecting the boundaries of people and their clothes for virtual try-on applications [29].

*Equal contribution.

†Inria, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France.

Code: https://github.com/rstrudel/segmenter

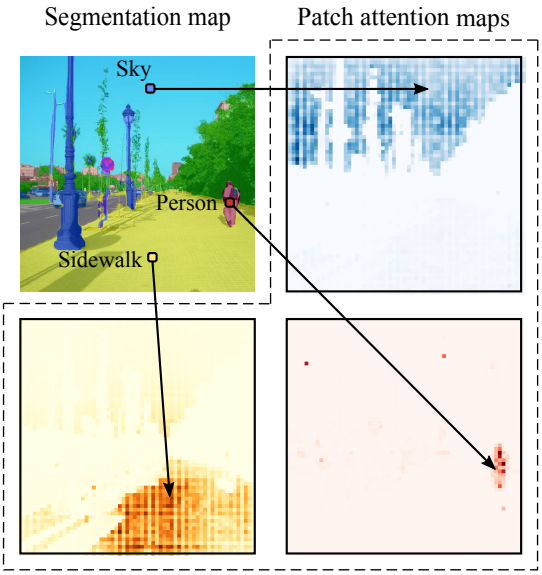


Figure 1: Our approach for semantic segmentation is purely transformer based. It leverages the global image context at every layer of the model. Attention maps from the first Segmenter layer are displayed for three 8×8 patches and highlight the early grouping of patches into semantically meaningful categories. The original image (top-left) is overlayed with segmentation masks produced by our method.

Despite much effort and large progress over recent years [10, 22, 31, 37, 48, 65, 66], image segmentation remains a challenging problem due to rich intra-class variation, context variation and ambiguities originating from occlusions and low image resolution.

Recent approaches to semantic segmentation typically rely on convolutional encoder-decoder architectures where the encoder generates low-resolution image features and the decoder upsamples features to segmentation maps with per-pixel class scores. State-of-the-art methods deploy Fully Convolutional Networks (FCN) [44] and achieve impressive results on challenging segmentation benchmarks [10, 23, 57, 58, 60, 64, 66]. These methods rely on learnable stacked convolutions that can capture semantically rich information and have been highly successful in computer vision. The local nature of convolutional filters, however,

limits the access to the global information in the image. Meanwhile, such information is particularly important for segmentation where the labeling of local patches often depends on the global image context. To circumvent this issue, DeepLab methods [8, 9, 10] introduce feature aggregation with dilated convolutions and spatial pyramid pooling. This allows to enlarge the receptive fields of convolutional networks and to obtain multi-scale features. Following recent progresses in NLP [50], several segmentation methods explore alternative aggregation schemes based on channel or spatial [22, 23, 61] attention and point-wise [66] attention to better capture contextual information. Such methods, however, still rely on convolutional backbones and are, hence, biased towards local interactions. An extensive use of specialised layers to remedy this bias [8, 10, 22, 58] suggests limitations of convolutional architectures for segmentation.

To overcome these limitations, we formulate the problem of semantic segmentation as a sequence-to-sequence problem and use a transformer architecture [50] to leverage contextual information at every stage of the model. By design, transformers can capture global interactions between elements of a scene and have no built-in inductive prior, see Figure 1. However, the modeling of global interactions comes at a quadratic cost which makes such methods prohibitively expensive when applied to raw image pixels [11]. Following the recent work on Vision Transformers (ViT) [19, 49], we split the image into patches and treat linear patch embeddings as input tokens for the transformer encoder. The contextualized sequence of tokens produced by the encoder is then upsampled by a transformer decoder to per-pixel class scores. For decoding, we consider either a simple point-wise linear mapping of patch embeddings to class scores or a transformer-based decoding scheme where learnable class embeddings are processed jointly with patch tokens to generate class masks. We conduct an extensive study of transformers for segmentation by ablating model regularization, model size, input patch size and its trade-off between accuracy and performance. Our Segmenter approach attains excellent results while remaining simple, flexible and fast. In particular, when using large models with small input patch size the best model reaches a mean IoU of 53.63% on the challenging ADE20K [68] dataset, surpassing all previous state-of-the-art convolutional approaches by a large margin of 5.3%. Such improvement partly stems from the global context captured by our method at every stage of the model as highlighted in Figure 1.

In summary, our work provides the following four contributions: (i) We propose a novel approach to semantic segmentation based on the Vision Transformer (ViT) [19] that does not use convolutions, captures contextual information by design and outperforms FCN based approaches. (ii) We present a family of models with varying levels of resolution which allows to trade-off between precision and runtime, ranging from *state-of-the-art* performance to models with fast inference and good performances. (iii) We propose a transformer-based decoder generating class masks which outperforms our linear baseline and can be extended to perform more general image segmentation tasks. (iv) We demonstrate that our approach yields *state-of-the-art* results on both ADE20K [68] and Pascal Context [38] datasets and is competitive on Cityscapes [14].

2. Related work

Semantic segmentation. Methods based on Fully Convolutional Networks (FCN) combined with encoder-decoder architectures have become the dominant approach to semantic segmentation. Initial approaches [21, 36, 39, 40] rely on a stack of consecutive convolutions followed by spatial pooling to perform dense predictions. Consecutive approaches [1, 4, 34, 41, 43] upsample high-level feature maps and combine them with low-level feature maps during decoding to both capture global information and recover sharp object boundaries. To enlarge the receptive field of convolutions in the first layers, several approaches [8, 33, 59] have proposed dilated or atrous convolutions. To capture global information in higher layers, recent work [9, 10, 65] employs spatial pyramid pooling to capture multi-scale contextual information. Combining these enhancements along with atrous spatial pyramid pooling, Deeplabv3+ [10] proposes a simple and effective encoder-decoder FCN architecture. Recent work [22, 23, 57, 58, 61, 66] replace coarse pooling by attention mechanisms on top of the encoder feature maps to better capture long-range dependencies.

While recent segmentation methods are mostly focused on improving FCN, the restriction to local operations imposed by convolutions may imply inefficient processing of global image context and suboptimal segmentation results. Hence, we propose a pure transformer architecture that captures global context at every layer of the model during the encoding and decoding stages.

Transformers for vision. Transformers [50] are now state of the art in many Natural Language Processing (NLP) tasks. Such models rely on self-attention mechanisms and capture long-range dependencies among tokens (words) in a sentence. In addition, transformers are well suited for parallelization, facilitating training on large datasets. The success of transformers in NLP has inspired several methods in computer vision combining CNNs with forms of self-attention to address object detection [7], semantic segmentation [53], panoptic segmentation [51], video processing [54] and few-shot classification [18].

Recently, the Vision Transformer (ViT) [19] introduced a convolution-free transformer architecture for image classification where input images are processed as sequences of patch tokens. While ViT requires training on very large datasets, DeiT [49] proposes a token-based distillation strategy and obtains a competitive vision transformer trained on the ImageNet-1k [16] dataset using a CNN as a

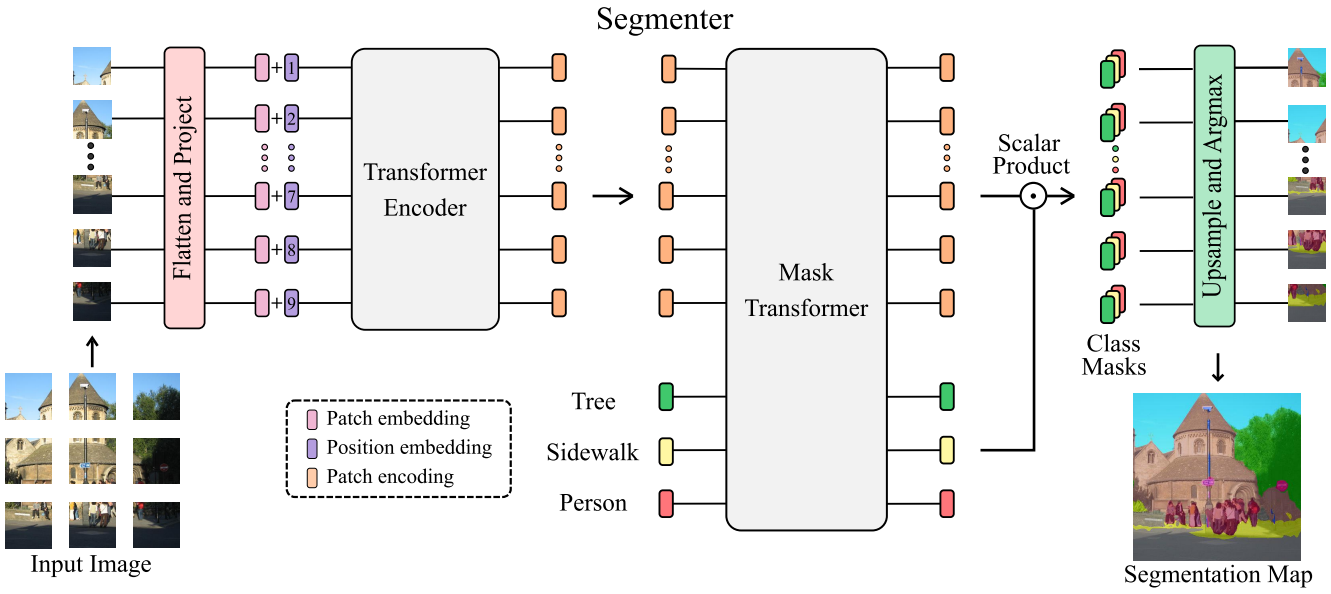


Figure 2: Overview of our approach Segmenter. (Left) Encoder: The image patches are projected to a sequence of embeddings and then encoded with a transformer. (Right) Decoder: A mask transformer takes as input the output of the encoder and class embeddings to predict segmentation masks. See text for details.

so we learn an "embedding" matrix E that is able to project each patch (that is a 1-dimensional vector of size C*P^2) into a vector of D dimension (the patch embedding)

teacher. Concurrent work extends this work to video classification [2, 6] and semantic segmentation [35, 67]. In more detail, SETR [67] uses a ViT backbone and a standard CNN decoder. Swin Transformer [35] uses a variant of ViT, composed of local windows, shifted between layers and UperNet as a pyramid FCN decoder.

Here, we propose Segmenter, a transformer encoder-decoder architecture for semantic image segmentation. Our approach relies on a ViT backbone and introduces a mask decoder inspired by DETR [7]. Our architecture does not use convolutions, captures global image context by design and results in competitive performance on standard image segmentation benchmarks.

3. Our approach: Segmenter

Segmenter is based on a fully transformer-based encoder-decoder architecture mapping a sequence of patch embeddings to pixel-level class annotations. An overview of the model is shown in Figure 2. The sequence of patches is encoded by a transformer encoder described in Section 3.1 and decoded by either a point-wise linear mapping or a mask transformer described in Section 3.2. Our model is trained end-to-end with a per-pixel cross-entropy loss. At inference time, argmax is applied after upsampling to obtain a single class per pixel.

3.1. Encoder

An image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ is split into a sequence of patches $\mathbf{x} = [x_1, ..., x_N] \in \mathbb{R}^{N \times P^2 \times C}$ where (P, P) is the patch size, $N = HW/P^2$ is the number of patches and C is the number of channels. Each patch is flattened into a 1D vector and then linearly projected to a patch embedding to produce a sequence of patch embeddings $\mathbf{x_0} = [\mathbf{E}x_1, ..., \mathbf{E}x_N] \in \mathbb{R}^{N \times D}$ where $\mathbf{E} \in \mathbb{R}^{D \times (P^2 C)}$. To capture positional information, learnable position embeddings $\mathbf{pos} = [\text{pos}_1, ..., \text{pos}_N] \in \mathbb{R}^{N \times D}$ are added to the sequence of patches to get the resulting input sequence of tokens $\mathbf{z_0} = \mathbf{x_0} + \mathbf{pos}$.

A transformer [50] encoder composed of L layers is applied to the sequence of tokens $\mathbf{z}_0$ to generate a sequence of contextualized encodings $\mathbf{z}_L \in \mathbb{R}^{N \times D}$. A transformer layer consists of a multi-headed self-attention (MSA) block followed by a point-wise MLP block of two layers with layer norm (LN) applied before every block and residual connections added after every block:

$$\mathbf{a_{i-1}} = \text{MSA}(\text{LN}(\mathbf{z_{i-1}})) + \mathbf{z_{i-1}}, \quad (1)$$

$$\mathbf{z_i} = \text{MLP}(\text{LN}(\mathbf{a_{i-1}})) + \mathbf{a_{i-1}}, \quad (2)$$

where $i \in \{1, ..., L\}$. The self-attention mechanism is composed of three point-wise linear layers mapping tokens to intermediate representations, queries $\mathbf{Q} \in \mathbb{R}^{N \times d}$, keys $\mathbf{K} \in \mathbb{R}^{N \times d}$ and values $\mathbf{V} \in \mathbb{R}^{N \times d}$. Self-attention is then computed as follows

$$\text{MSA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}. \quad (3)$$

The transformer encoder maps the input sequence $\mathbf{z_0} = [z_{0,1}, ..., z_{0,N}]$ of embedded patches with position encoding to $\mathbf{z_L} = [z_{L,1}, ..., z_{L,N}]$, a contextualized encoding sequence containing rich semantic information used by the decoder. In the following section we introduce the decoder.

x is now a vector of N flattened patches of dimension P^2*C

3.2. Decoder

The sequence of patch encodings $\mathbf{z_L} \in \mathbb{R}^{N\times D}$ is decoded to a segmentation map $\mathbf{s} \in \mathbb{R}^{H\times W\times K}$ where K is the number of classes. The decoder learns to map patch-level encodings coming from the encoder to patch-level class scores. Next these patch-level class scores are upsampled by bilinear interpolation to pixel-level scores. We describe in the following a linear decoder, which serves as a baseline, and our approach, a mask transformer, see Figure 2.

Linear. A point-wise linear layer is applied to the patch encodings $\mathbf{z_L} \in \mathbb{R}^{N\times D}$ to produce patch-level class logits $\mathbf{z_{lin}} \in \mathbb{R}^{N\times K}$. The sequence is then reshaped into a 2D feature map $\mathbf{s_{lin}} \in \mathbb{R}^{H/P\times W/P\times K}$ and bilinearly upsampled to the original image size $\mathbf{s} \in \mathbb{R}^{H\times W\times K}$. A softmax is then applied on the class dimension to obtain the final segmentation map.

Mask Transformer. For the transformer-based decoder, we introduce a set of K learnable class embeddings $\mathbf{cls} = [\text{cls}_1, ..., \text{cls}_K] \in \mathbb{R}^{K\times D}$ where K is the number of classes. Each class embedding is initialized randomly and assigned to a single semantic class. It will be used to generate the class mask. The class embeddings $\mathbf{cls}$ are processed jointly with patch encodings $\mathbf{z_L}$ by the decoder as depicted in Figure 2. The decoder is a transformer encoder composed of M layers. Our mask transformer generates K masks by computing the scalar product between L2-normalized patch embeddings $\mathbf{z'_M} \in \mathbb{R}^{N\times D}$ and class embeddings $\mathbf{c} \in \mathbb{R}^{K\times D}$ output by the decoder. The set of class masks is computed as follows

$$\text{Masks}(\mathbf{z'_M}, \mathbf{c}) = \mathbf{z'_M}\mathbf{c}^T \tag{4}$$

where $\text{Masks}(\mathbf{z'_M}, \mathbf{c}) \in \mathbb{R}^{N\times K}$ is a set of patch sequence. Each mask sequence is then reshaped into a 2D mask to form $\mathbf{s_{mask}} \in \mathbb{R}^{H/P\times W/P\times K}$ and bilinearly upsampled to the original image size to obtain a feature map $\mathbf{s} \in \mathbb{R}^{H\times W\times K}$. A softmax is then applied on the class dimension followed by a layer norm to obtain pixel-wise class score forming the final segmentation map. The masks sequences are softly exclusive to each other i.e. $\sum_{k=1}^{K} s_{i,j,k} = 1$ for all $(i, j) \in H \times W$.

Our mask transformer is inspired by DETR [7], MaxDeepLab [52] and SOLO-v2 [55] which introduce object embeddings [7] to produce instance masks [52, 55]. However, unlike our method, MaxDeep-Lab uses an hybrid approach based on CNNs and transformers and splits the pixel and class embeddings into two streams because of computational constraints. Using a pure transformer architecture and leveraging patch level encodings, we propose a simple approach that processes the patch and class embeddings jointly during the decoding phase. Such approach allows to produce dynamical filters, changing with the input. While we address semantic segmentation in this work, our mask transformer can also be directly adapted to perform panoptic segmentation by replacing the class embeddings by object embeddings.

Model	Backbone	Layers	Token size	Heads	Params
Seg-Ti	ViT-Ti	12	192	3	6M
Seg-S	ViT-S	12	384	6	22M
Seg-B	ViT-B	12	768	12	86M
Seg-B†	DeiT-B	12	768	12	86M
Seg-L	ViT-L	24	1024	16	307M

Table 1: Details of Transformer variants.

4. Experimental results

4.1. Datasets and metrics

ADE20K [68]. This dataset contains challenging scenes with fine-grained labels and is one of the most challenging semantic segmentation datasets. The training set contains 20,210 images with 150 semantic classes. The validation and test set contain 2,000 and 3,352 images respectively.

Pascal Context [38]. The training set contains 4,996 images with 59 semantic classes plus a background class. The validation set contains 5,104 images.

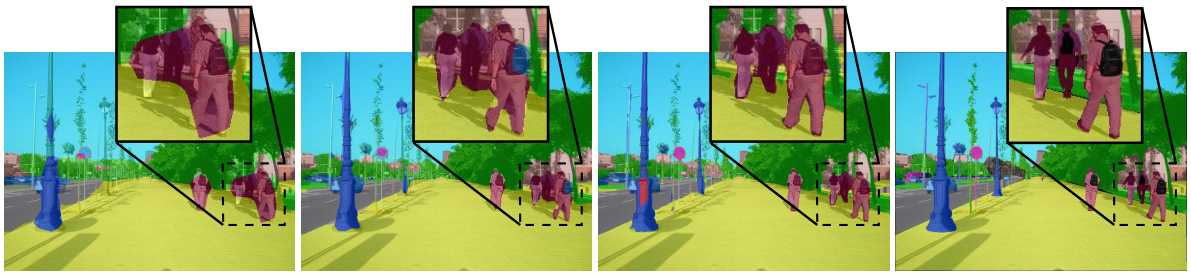
Cityscapes [14]. The dataset contains 5,000 images from 50 different cities with 19 semantic classes. There are 2,975 images in the training set, 500 images in the validation set and 1,525 images in the test set.

Metrics. We report Intersection over Union (mIoU) averaged over all classes.

4.2. Implementation details

Transformer models. For the encoder, we build upon the vision transformer ViT [19] and consider "Tiny", "Small", "Base" and "Large" models described in Table 1. The parameters varying in the transformer encoder are the number of layers and the token size. The head size of a multi-headed self-attention (MSA) block is fixed to 64, the number of heads is the token size divided by the head size and the hidden size of the MLP following MSA is four times the token size. We also use DeiT [49], a variant of the vision transformer. We consider models representing the image at different resolutions and use input patch sizes 8×8, 16×16 and 32×32. In the following, we use an abbreviation to describe the model variant and patch size, for instance Seg-B/16 denotes the "Base" variant with 16×16 input patch size. Models based on DeiT are denoted with a †, for instance Seg-B†/16.

ImageNet pre-training. Our Segmenter models are pre-trained on ImageNet, ViT is pre-trained on ImageNet-21k with strong data augmentation and regularization [47] and its variant DeiT is pre-trained on ImageNet-1k. The original ViT models [19] have been trained with random cropping only, whereas the training procedure proposed by [47] uses a combination of dropout [46] and stochastic depth [30] as regularization and Mixup [62] and RandAugment [15] as data augmentations. This significantly improves the ImageNet top-1 accuracy, i.e., it obtains a gain of +2% on ViT-B/16. We fine-tuned ViT-B/16 on ADE20K with



(a) Patch size 32×32 (b) Patch size 16×16 (c) Patch size 8×8 (d) Ground Truth

Figure 3: Impact of the model patch size on the segmentation maps.

models from [19] and [47] and observe a significant difference, namely a mIoU of 45.69% and 48.06% respectively. In the following, all the Segmenter models will be initialized with the improved ViT models from [47]. We use publicly available models provided by the image classification library timm [56] and Google research [20]. Both models are pre-trained at an image resolution of 224 and fine-tuned on ImageNet-1k at a resolution of 384, except for ViT-B/8 which has been fine-tuned at a resolution of 224. We keep the patch size fixed and fine-tune the models for the semantic segmentation task at higher resolution depending on the dataset. As the patch size is fixed, increasing resolution results in longer token sequences. Following [19], we bilinearly interpolate the pre-trained position embeddings according to their original position in the image to match the fine-tuning sequence length. The decoders, described in Section 3.2 are initialized with random weights from a truncated normal distribution [25].

Data augmentation. During training, we follow the standard pipeline from the semantic segmentation library MMSegmentation [13], which does mean substraction, random resizing of the image to a ratio between 0.5 and 2.0 and random left-right flipping. We randomly crop large images and pad small images to a fixed size of 512×512 for ADE20K, 480×480 for Pascal-Context and 768×768 for Cityscapes. On ADE20K, we train our largest model Seg-L-Mask/16 with a resolution of 640×640, matching the resolution used by the Swin Transformer [35].

Optimization. To fine-tune the pre-trained models for the semantic segmentation task, we use the standard pixel-wise cross-entropy loss without weight rebalancing. We use stochastic gradient descent (SGD) [42] as the optimizer with a base learning rate γ_0 and set weight decay to 0. Following the seminal work of DeepLab [33] we adopt the "poly" learning rate decay $\gamma = \gamma_0(1 - \frac{N_{iter}}{N_{total}})^{0.9}$ where N_{iter} and N_{total} represent the current iteration number and the total iteration number. For ADE20K, we set the base learning rate γ_0 to 10^{-3} and train for 160K iterations with a batch size of 8. For Pascal Context, we set γ_0 to 10^{-3} and train for 80K iterations with a batch size of 16. For Cityscapes, we set γ_0 to 10^{-2} and train for 80K iterations with a batch size of 8. The schedule is similar to DeepLabv3+ [10] with learning rates divided by a factor 10 except for Cityscapes where we use a factor of 1.

		Stochastic Depth		
		0.0	0.1	0.2
Dropout	0.0	45.01	**45.37**	45.10
	0.1	42.02	42.30	41.14
	0.2	36.49	36.63	35.67

Table 2: Mean IoU comparison of different regularization schemes using Seg-S/16 on ADE20K validation set.

Method	Backbone	Patch size	Im/sec	ImNet acc.	mIoU (SS)
Seg-Ti/16	ViT-Ti	16	396	78.6	39.03
Seg-S/32	ViT-S	32	1032	80.5	40.64
Seg-S/16	ViT-S	16	196	83.7	45.37
Seg-B†/16	DeiT-B	16	92	85.2	47.08
Seg-B/32	ViT-B	32	516	83.3	43.07
Seg-B/16	ViT-B	16	92	86.0	48.06
Seg-B/8	ViT-B	8	7	85.7	49.54
Seg-L/16	ViT-L	16	33	87.1	50.71

Table 3: Performance comparison of different Segmenter models with varying backbones and input patch sizes on ADE20K validation set.

Inference. To handle varying image sizes during inference, we use a sliding-window with a resolution matching the training size. For multi-scale inference, following standard practice [10] we use rescaled versions of the image with scaling factors of (0.5, 0.75, 1.0, 1.25, 1.5, 1.75) and left-right flipping and average the results.

4.3. Ablation study

In this section, we ablate different variants of our approach on the ADE20K validation set. We investigate model regularization, model size, patch size, model performance, training dataset size, compare Segmenter to convolutional approaches and evaluate different decoders. Unless stated otherwise, we use the baseline linear decoder and report results using single-scale inference.

Regularization. We first compare two forms of regularization, dropout [46] and stochastic depth [30], and show that stochastic depth consistently improves transformer training for segmentation. CNN models rely on batch nor-

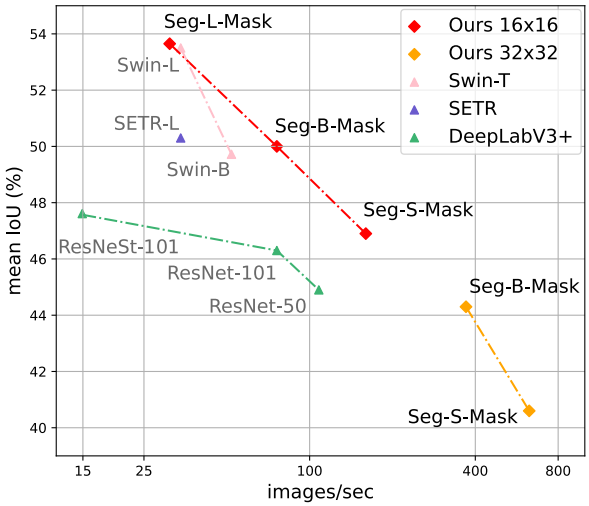


Figure 4: Images per second and mean IoU for our approach compared to other methods on ADE20K validation set. Segmenter models offer a competitive trade-off in terms of performance and precision.

malization [32] which also acts as a regularizer. In contrast, transformers are usually composed of layer normalization [3] combined with dropout as a regularizer during training [17, 19]. Dropout randomly ignores tokens given as input of a block and stochastic depth randomly skips a learnable block of the model during the forward pass. We compare regularizations on Seg-S/16 based on ViT-S/16 backbone. Table 2 shows that stochastic depth set to 0.1, dropping 10% of the layers randomly, consistently improves the performance, with 0.36% when the dropout is set to 0 compared to the baseline without regularization. Dropout consistently hurts performances, either alone or when combined with stochastic depth. This is consistent with [49] which observed the negative impact of dropout for image classification. From now on, all the models will be trained with stochastic depth set to 0.1 and without dropout.

Transformer size. We now study the impact of transformers size on performance by varying the number of layers and the tokens size for a fixed patch size of 16. Table 3 shows that performance scales nicely with the backbone capacity. When doubling the token dimension, from Seg-S/16 to Seg-B/16, we get a 2.69% improvement. When doubling the number of layers, from Seg-B/16 to Seg-L/16, we get an improvement of 2.65%. Finally, our largest Segmenter model, Seg-L/16, achieves a strong mIoU of 50.71% with a simple decoding scheme on the ADE20K validation dataset with single scale inference. The absence of tasks-specific layers vastly used in FCN models suggests that transformer based methods provide more expressive models, well suited for semantic segmentation.

Patch size. Representing an image with a patch sequence provides a simple way to trade-off between speed and accuracy by varying the patch size. While increasing the patch size results in a coarser representation of the image, it results in a smaller sequence that is faster to process. The third and fourth parts of Table 3 report the performance for ViT backbones and varying patch sizes. We observe that the patch size is a key factor for semantic segmentation performance. It is similarly important to the model size. Indeed, going from a patch size 32 to 16 we observe an improvement of 5% for Seg-B. For Seg-B, we also report results for a patch size of 8 and report an mIoU of 49.54%, reducing the gap from ViT-B/8 to ViT-L/16 to 1.17% while requiring substantially fewer parameters. This trend shows that reducing the patch size is a robust source of improvement which does not introduce any parameters but requires to compute attention over longer sequences, increasing the compute time and memory footprint. If it was computationally feasible, ViT-L/8 would probably be the best performing model. Going towards more computation and memory efficient transformers handling larger sequence of smaller patches is a promising direction.

To further study the impact of patch size, we show segmentation maps generated by Segmenter models with decreasing patch size in Figure 3. We observe that for a patch size of 32, the model learns a globally meaningful segmentation but produces poor boundaries, for example the two persons on the left are predicted by a single blob. Reducing the patch size leads to considerably sharper boundaries as can be observed when looking at the contours of persons. Hard to segment instances as the thin streetlight pole in the background are only captured at a resolution of 8. In Table 4, we report mean IoU with respect to the

Method	Decoder	Small	Medium	Large	mIoU (SS)
DeepLab RNeSt-101	UNet	37.85	50.89	50.67	46.47
Seg-B/32	Linear	31.95	47.82	49.44	43.07
Seg-B-Mask/32	Mask	32.29	49.44	50.82	44.19
Seg-B†/16	Linear	38.31	50.91	52.08	47.10
Seg-B†-Mask/16	Mask	40.49	51.37	54.24	48.70
Seg-B/16	Linear	39.57	51.32	53.28	48.06
Seg-B-Mask/16	Mask	40.16	52.61	52.66	48.48
Seg-B/8	Linear	41.43	54.35	52.85	49.54
Seg-L/16	Linear	42.08	54.67	55.39	50.71
Seg-L-Mask/16	Mask	42.02	54.83	57.06	51.30

Table 4: Evaluation with respect to the object size on ADE20k validation set (mean IoU). Comparison of DeepLabv3+ ResNeSt-101 to Segmenter models with a linear or a mask transformer decoder.

Dataset Size	4k	8k	12k	16k	20k
mIoU (SS)	38.31	41.87	43.42	44.61	45.37

Table 5: Performance comparison of Seg-S/16 models trained with increasing dataset size and evaluated on ADE20K validation set.

object size and compare Segmenter to DeepLabv3+ with ResNeSt backbone. To reproduce DeepLabv3+ results, we used models from the MMSegmentation library [13]. We observe how Seg-B/8 improvement over Seg-B/16 comes mostly from small and medium instances with a gain of 1.27% and 1.74% respectively. Also, we observe that overall the biggest improvement of Segmenter over DeepLab comes from large instances where Seg-L-Mask/16 shows an improvement of 6.39%.

Decoder variants. In this section, we compare different decoder variants. We evaluate the mask transformer introduced in Section 3.2 and compare it to the linear baseline. The mask transformer has 2 layers with the same token and hidden size as the encoder. Table 4 reports the mean IoU performance. The mask transformer provides consistent improvements over the linear baseline. The most significant gain of 1.6% is obtained for Seg-B†/16, for Seg-B-Mask/32 we obtain a 1.1% improvement and for Seg-L/16 a gain of 0.6%. In Table 4 we also examine the gain of different models with respect to the object size. We observe gains both on small and large objects, showing the benefit of using dynamical filters. In most cases the gain is more significant for large objects, i.e., 1.4% for Seg-B/32, 2.1% for Seg-B†/16 and and 1.7% for Seg-L/16. The class embeddings learned by the mask transformer are semantically meaningful, i.e., similar classes are nearby, see Figure 8 for more details.

Transformer versus FCN. Table 4 and Table 6 compare our approach to FCN models and DeepLabv3+ [10] with ResNeSt backbone [63], one of the best fully-convolutional approaches. Our transformer approach provides a significant improvement over this state-of-the-art convolutional approach, highlighting the ability of transformers to capture global scene understanding. Segmenter consistently outperforms DeepLab on large instances with an improvement of more than 4% for Seg-L/16 and 6% for Seg-L-Mask/16. However, DeepLab performs similarly to Seg-B/16 on small and medium instances while having a similar number of parameters. Seg-B/8 and Seg-L/16 perform best on small and medium instances though at higher computational cost.

Performance. In Figure 4, we compare our models to several *state-of-the-art* methods in terms of images per seconds and mIoU and show a clear advantage of Segmenter over FCN based models (green curve). We also show that our approach compares favorably to recent transformer based approach, our largest model Seg-L-Mask/16 is on-par with Swin-L and outperforms SETR-MLA. We observe that Seg/16 models perform best in terms of accuracy versus compute time with Seg-B-Mask/16 offering a good trade-off. Seg-B-Mask/16 outperforms FCN based approaches with similar inference speed, matches SETR-MLA while being twice faster and requiring less parameters and outperforms Swin-B both in terms of inference speed and performance. Seg/32 models learn coarser segmentation maps as discussed in the previous section and enable fast inference with 400 images per second for Seg-B-Mask/32, four times faster than ResNet-50 while providing similar performances. To compute the images per second, we use a V100 GPU, fix the image resolution to 512 and for each model we maximize the batch size allowed by memory for a fair comparison.

Method	Backbone	Im/sec	mIoU	+MS
OCR [60]	HRNetV2-W48	83	-	45.66
ACNet [24]	ResNet-101	-	-	45.90
DNL [57]	ResNet-101	-	-	45.97
DRANet [22]	ResNet-101	-	-	46.18
CPNet [58]	ResNet-101	-	-	46.27
DeepLabv3+ [10]	ResNet-101	76	45.47	46.35
DeepLabv3+ [10]	ResNeSt-101	15	46.47	47.27
DeepLabv3+ [10]	ResNeSt-200	-	-	48.36
SETR-L MLA [67]	ViT-L/16	34	48.64	50.28
Swin-L UperNet [35]	Swin-L/16	34	52.10	**53.50**
Seg-B†/16	DeiT-B/16	77	47.08	48.05
Seg-B†-Mask/16	DeiT-B/16	76	48.70	50.08
Seg-L/16	ViT-L/16	33	50.71	52.25
Seg-L-Mask/16	ViT-L/16	31	51.82	**53.63**

Table 6: State-of-the-art comparison on ADE20K validation set.

Dataset size. Vision Transformers highlighted the importance of large datasets to attain good performance for the task of image classification. At the scale of a semantic segmentation dataset, we analyze Seg-S/16 performance on ADE20k dataset in Table 5 when trained with a dataset of increasing size. We observe an important drop in performance when the training set size is below 8k images. This shows that even during fine-tuning transformers performs best with a sufficient amount of data.

4.4. Comparison with state of the art

In this section, we compare the performance of Segmenter with respect to the state-of-the-art methods on ADE20K, Pascal Context and Cityscapes datasets.

ADE20K. Seg-B†/16 pre-trained on ImageNet-1k matches the *state-of-the-art* FCN method DeepLabv3+ ResNeSt-200 [63] as shown in Table 6. Adding our mask transformer, Seg-B†-Mask/16 improves by 2% and achieves a 50.08% mIoU, outperforming all FCN methods. Our best model, Seg-L-Mask/16 attains a state-of-the-art performance of 53.63%, outperforming by a margin of 5.27% mIoU DeepLabv3+ ResNeSt-200 and the transformer-based methods SETR [67] and Swin-L UperNet [35].

Pascal Context Table 7 reports the performance on Pascal Context. Seg-B†models are competitive with FCN methods and the larger Seg-L/16 model already provides *state-of-the-art* performance, outperforming SETR-L. Performances can be further enhanced with our mask transformer, Seg-L-Mask/16, improving over the linear decoder by 2.5% and achieving a performance of 59.04% mIoU. In particular, we report an improvement of 2.8% over OCR

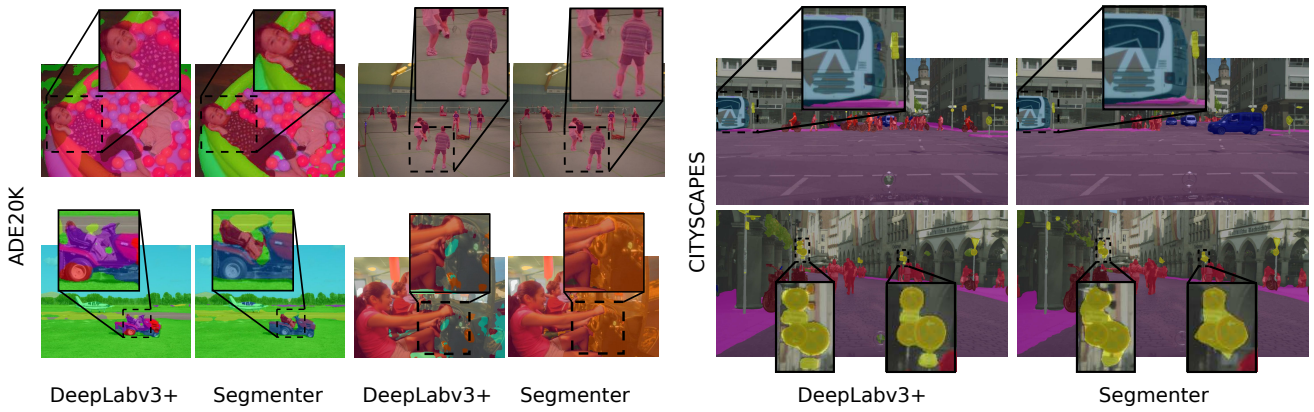


Figure 5: Qualitative comparison of Seg-L-Mask/16 performance with DeepLabV3+ ResNeSt-101. See Section C for additional qualitative results.

Method	Backbone	mIoU (MS)
DeepLabv3+ [10]	ResNet-101	48.5
DANet [23]	ResNet-101	52.6
ANN [69]	ResNet101	52.8
CPNet [58]	ResNet-101	53.9
CFNet [64]	ResNet-101	54.0
ACNet [24]	ResNet-101	54.1
APCNet [26]	ResNet101	54.7
DNL [57]	HRNetV2-W48	55.3
DRANet [22]	ResNet-101	55.4
OCR [60]	HRNetV2-W48	56.2
SETR-L MLA [67]	ViT-L/16	55.8
Seg-B†/16	DeiT-B/16	53.9
Seg-B†-Mask/16	DeiT-B/16	55.0
Seg-L/16	ViT-L/16	56.5
Seg-L-Mask/16	ViT-L/16	**59.0**

Table 7: State-of-the-art comparison on Pascal Context validation set.

Method	Backbone	mIoU (MS)
PSANet [66]	ResNet-101	79.1
DeepLabv3+ [10]	Xception-71	79.6
ANN [69]	ResNet-101	79.9
MDEQ [5]	MDEQ	80.3
DeepLabv3+ [10]	ResNeSt-101	80.4
DNL [57]	ResNet-101	80.5
CCNet [31]	ResNet-101	81.3
Panoptic-Deeplab [12]	Xception-71	81.5
DeepLabv3+ [10]	ResNeSt-200	**82.7**
SETR-L PUP [67]	ViT-L/16	**82.2**
Seg-B†/16	DeiT-B/16	80.5
Seg-B†-Mask/16	DeiT-B/16	80.6
Seg-L/16	ViT-L/16	80.7
Seg-L-Mask/16	ViT-L/16	81.3

Table 8: State-of-the-art comparison on Cityscapes validation set.

HRNetV2-W48 and 3.2% over SETR-L MLA.

Cityscapes. Table 8 reports the performance of Segmenter on Cityscapes. We use a variant of mask transformer for Seg-L-Mask/16 with only one layer in the decoder as two layers did not fit into memory due to the large input resolution of 768×768. Both Seg-B and Seg-L methods are competitive with other *state-of-the-art methods* with Seg-L-Mask/16 achieving a mIoU of 81.3%.

Qualitative results. Figure 5 shows a qualitative comparison of Segmenter and DeepLabv3+ with ResNeSt backbone, for which models were provided by the MMSegmentation [13] library. We can observe that Deeplabv3+ tends to generate sharper object boundaries while Segmenter provides more consistent labels on large instances and handles partial occlusions better.

5. Conclusion

This paper introduces a pure transformer approach for semantic segmentation. The encoding part builds up on the recent Vision Transformer (ViT), but differs in that we rely on the encoding of all images patches. We observe that the transformer captures the global context very well. Applying a simple point-wise linear decoder to the patch encodings already achieves excellent results. Decoding with a mask transformer further improves the performance. We believe that our end-to-end encoder-decoder transformer is a first step towards a unified approach for semantic segmentation, instance segmentation and panoptic segmentation.

6. Acknowledgements

We thank Andreas Steiner for providing the ViT-Base model trained on 8×8 patches and Gauthier Izacard for the helpful discussions. This work was partially supported by the HPC resources from GENCI-IDRIS (Grant 2020-AD011011163R1), the Louis Vuitton ENS Chair on Artificial Intelligence, and the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

References

[1] Md Amirul Islam, Mrigank Rochan, Neil D. B. Bruce, and Yang Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, 2017. 2

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. ViViT: A video vision transformer. In *ICCV*, 2021. 3

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint*, 2016. 6

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017. 2

[5] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale deep equilibrium models. In *NeurIPS*, 2020. 8

[6] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint*, 2021. 3

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with Transformers. In *ECCV*, 2020. 2, 3, 4

[8] 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. *TPAMI*, 2018. 2

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

[10] 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. 1, 2, 5, 7, 8

[11] Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, David Luan, and Ilya Sutskever. Generative pretraining from pixels. *PLMR*, 2020. 2

[12] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-Deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 8

[13] MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 5, 7, 8

[14] 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. 2, 4

[15] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 3008–3017. IEEE, 2020. 4

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 6

[18] Carl Doersch, Ankush Gupta, and Andrew Zisserman. CrossTransformers: spatially-aware few-shot transfer. In *NeurIPS*, 2020. 2

[19] 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, 4, 5, 6

[20] 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. Vision transformer models. https://console.cloud.google.com/storage/browser/vit_models, 2021. 5

[21] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *TPAMI*, 2013. 2

[22] J. Fu, J. Liu, J. Jiang, Y. Li, Y. Bao, and H. Lu. Scene segmentation with Dual Relation-Aware attention Network. *TNNLS*, 2020. 1, 2, 7, 8

[23] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual Attention Network for scene segmentation. In *CVPR*, 2019. 1, 2, 8

[24] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive Context Network for scene parsing. *ICCV*, 2019. 7, 8

[25] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *NeurIPS*, 2018. 5

[26] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive Pyramid Context Network for semantic segmentation. In *CVPR*, 2019. 8

[27] Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He, and Paul Kennedy. Deep learning techniques for medical image segmentation: Achievements and challenges. *JDI*, 2019. 1

[28] Zhang-Wei Hong, Chen Yu-Ming, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, Hsuan-Kung Yang, Brian Hsi-Lin Ho, Chih-Chieh Tu, Yueh-Chuan Chang, Tsu-Ching Hsiao, Hsin-Wei Hsiao, Sih-Pin Lai, and Chun-Yi Lee. Virtual-to-real: Learning to control in visual semantic segmentation. *IJCAI*, 2018. 1

[29] Chia-Wei Hsieh, Chieh-Yun Chen, Chien-Lung Chou, Hong-Han Shuai, Jiaying Liu, and Wen-Huang Cheng. Fashionon: Semantic-guided image-based virtual try-on with detailed human and clothing information. In *ACM MM*, 2019. 1

[30] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 4, 5

[31] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. CCNet: Criss-Cross attention for semantic segmentation. In *ICCV*, 2019. 1, 8

[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[33] Chen Liang-Chieh, George Papandreou, Iasonas Kokkinos, kevin murphy, and Alan Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 2, 5

[34] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 2

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint*, 2021. 3, 5, 7

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for semantic segmentation. In *CVPR*, 2015. 2

[37] Shervin Minaee, Yuri Boykov, F. Porikli, Antonio J. Plaza, N. Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *TPAMI*, 2021. 1

[38] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014. 2, 4

[39] F. Ning, D. Delhomme, Yann LeCun, F. Piano, Léon Bottou, and Paolo Emilio Barbano. Toward automatic phenotyping of developing embryos from videos. *TIP*, 2005. 2

[40] Pedro Pinheiro and Ronan Collobert. Recurrent Convolutional Neural Networks for scene labeling. In *ICML*, 2014. 2

[41] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017. 2

[42] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 1951. 5

[43] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[44] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully Convolutional Networks for semantic segmentation. *TPAMI*, 2017. 1

[45] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges. In *ITSC*, 2017. 1

[46] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 4, 5

[47] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. *arXiv preprint*, 2021. 4, 5

[48] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 2020. 1

[49] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training Data-Efficient image Transformers and distillation through attention. *arXiv preprint*, 2020. 2, 4, 6

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2, 3

[51] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan L. Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint*, 2020. 2

[52] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan L. Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint*, 2020. 4

[53] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-Deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020. 2

[54] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2

[55] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *NeurIPS*, 2020. 4

[56] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2020. 5

[57] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled Non-local Neural Networks. In *ECCV*, 2020. 1, 2, 7, 8

[58] Changqian Yu, Jingbo Wang, Changxin Gao, Gang Yu, Chunhua Shen, and Nong Sang. Context Prior for scene segmentation. In *CVPR*, 2020. 1, 2, 7, 8

[59] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 2

[60] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-Contextual Representations for semantic segmentation. In *ECCV*, 2020. 1, 7, 8

[61] Yuhui Yuan and Jingdong Wang. OCNet: Object Context Network for scene parsing. *arXiv preprint*, 2018. 2

[62] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 4

[63] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander J. Smola. Resnest: Split-attention networks. *arXiv preprint*, 2020. 7

[64] H. Zhang, H. Zhang, C. Wang, and J. Xie. Co-occurrent features in semantic segmentation. In *CVPR*, 2019. 1, 8

[65] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. In *CVPR*, 2017. 1, 2

[66] 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 *ECCV*, September 2018. 1, 2, 8

[67] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic

segmentation from a sequence-to-sequence perspective with Transformers. *arXiv preprint*, 2020. 3, 7, 8

[68] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *IJCV*, 2019. 2, 4

[69] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *ICCV*, 2019. 8

Appendix

This appendix presents additional results. We study the impact of ImageNet pretraining on the performance and demonstrate its importance in Section A. To gain more insight about our approach Segmenter, we analyze its attention maps and the learned class embeddings in Section B. Finally, we give an additional qualitative comparison of Segmenter to DeepLabv3+ on ADE20K, Cityscapes and Pascal Context in Section C.

A. ImageNet pre-training

To study the impact of ImageNet pre-training on Segmenter, we compare our model pre-trained on ImageNet with equivalent models trained from scratch. To train from scratch, the weights of the model are initialized randomly with a truncated normal distribution. We use a base learning rate of 10^{-3} and two training procedures. First, we follow the fine-tuning procedure and use SGD optimizer with "poly" scheduler. Second, we follow a more standard procedure when training a transformer from scratch where we use AdamW with a cosine scheduler and a linear warmup for $16K$ iterations corresponding to 10% of the total number of iterations. Table 9 reports results for Seg-S/16. We observe that when pre-trained on ImageNet-21k using SGD, Seg-S/16 reaches 45.37% yielding a 32.9% improvement over the best randomly initialized model.

Method	Pre-training	Optimizer	mIoU (SS)
Seg-S/16	None	AdamW	4.42
Seg-S/16	None	SGD	12.51
Seg-S/16	ImageNet-21k	AdamW	34.77
Seg-S/16	ImageNet-21k	SGD	**45.37**

Table 9: Impact of pretraining on the performance on ADE20K validation set.

B. Attention maps and class embeddings

To better understand how our approach Segmenter processes images, we display attention maps of Seg-B/8 for 3 images in Figure 6. We resize attention maps to the original image size. For each image, we analyze attention maps of a patch on a small instance, for example lamp, cow or car. We also analyze attention maps of a patch on a large instance, for example bed, grass and road. We observe that the attention map field-of-view adapts to the input image and the instance size, gathering global information on large instances and focusing on local information on smaller instances. This adaptability is typically not possible with CNN which have a constant field-of-view, independently of the data. We also note there is progressive gathering of information from bottom to top layers, as for example on the cow instance, where the model first identifies the cow the patch belongs to, then identifies other cow instances. We observe that attention maps of lower layers depends strongly on the selected patch while they tend to be more similar for higher layers.

Additionally, to illustrate the larger receptive field size of Segmenter compared to CNNs, we reported the size of the attended area in Figure 7, where each dot shows the mean attention distance for one of the 12 attention heads at each layer. Already for the first layer, some heads attend to distant patches which clearly lie outside the receptive field of ResNet/ResNeSt initial layers.

To gain some understanding of the class embeddings learned with the mask transformer, we project embeddings into 2D with a singular value decomposition. Figure 8 shows that these projections group instances such as means of transportation (bottom left), objects in a house (top) and outdoor categories (middle right). It displays an implicit clustering of semantically related categories.

C. Qualitative results

We present additional qualitative results including comparison with DeepLabv3+ ResNeSt-101 and failure cases in Figures 9, 10 and 11. We can see in Figure 9 that Segmenter produces more coherent segmentation maps than DeepLabv3+. This is the case for the wedding dress in (a) or the airplane signalmen's helmet in (b). In Figure 10, we show how for some examples, different segments which look very similar are confused both in DeepLabv3+ and Segmenter. For example, the armchairs and couchs in (a), the cushions and pillows in (b) or the trees, flowers and plants in (c) and (d). In Figure 11, we can see how DeepLabv3+ handles better the boundaries between different people entities. Finally, both Segmenter and DeepLabv3+ have problems segmenting small instances such as lamp, people or flowers in Figure 12 (a) or the cars and signals in Figure 12 (b).

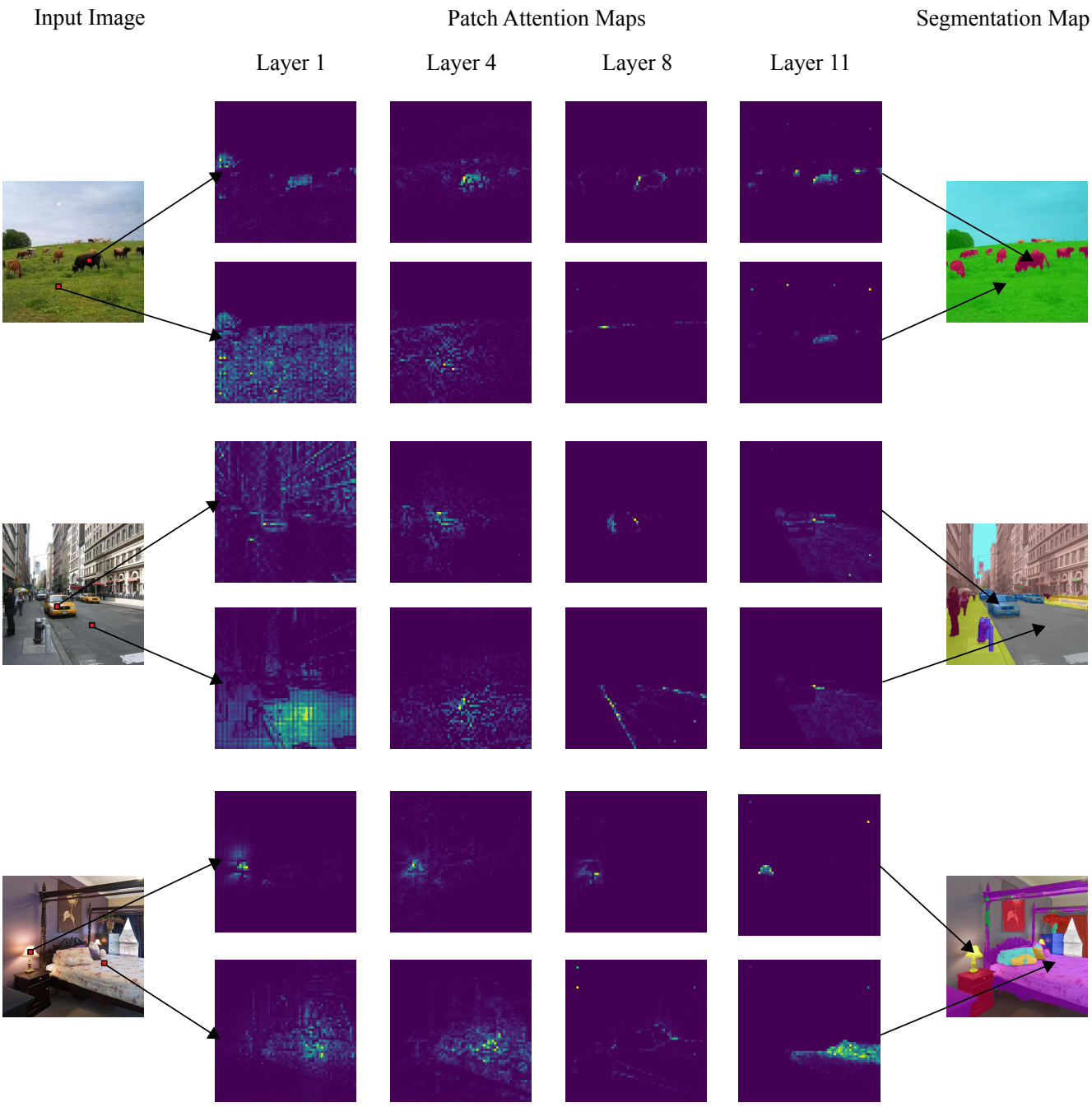


Figure 6: Seg-B/8 patch attention maps for the layers 1, 4, 8 and 11.

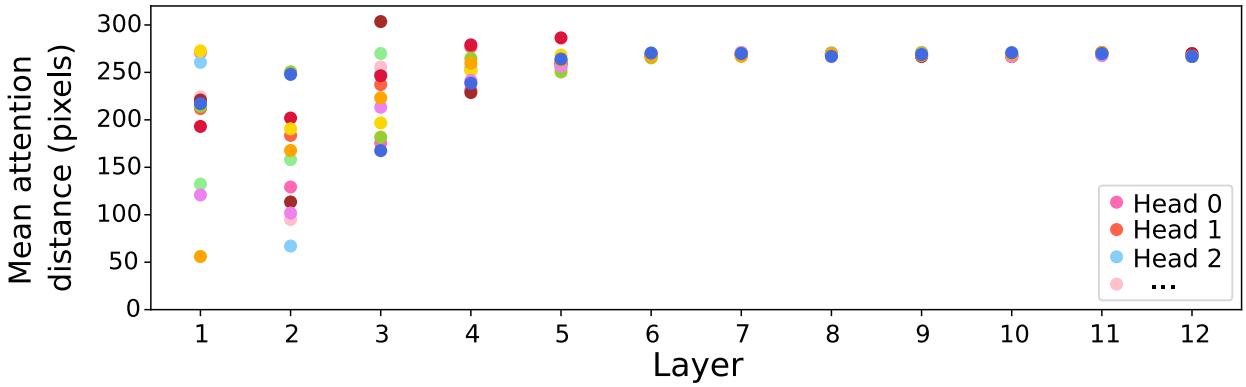


Figure 7: Size of attended area by head and model depth.

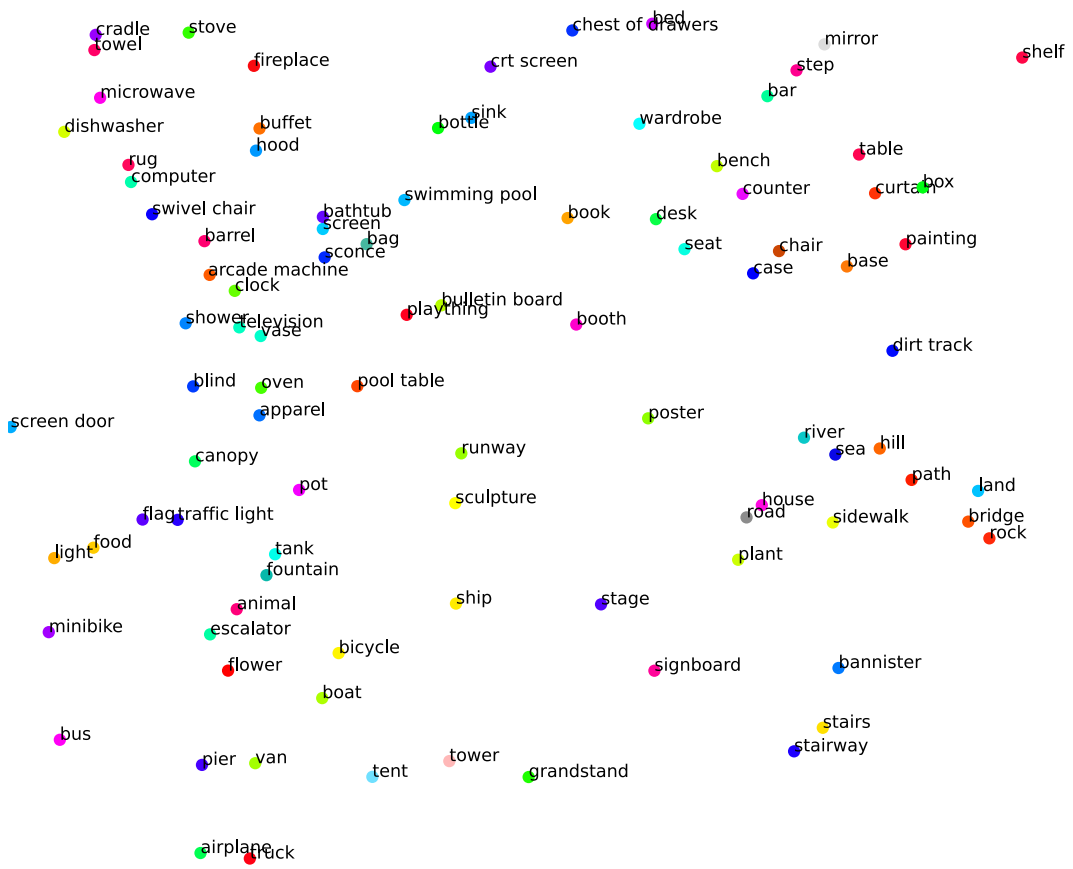


Figure 8: Singular value decompostion of the class embeddings learned with the mask transformer on ADE20K.

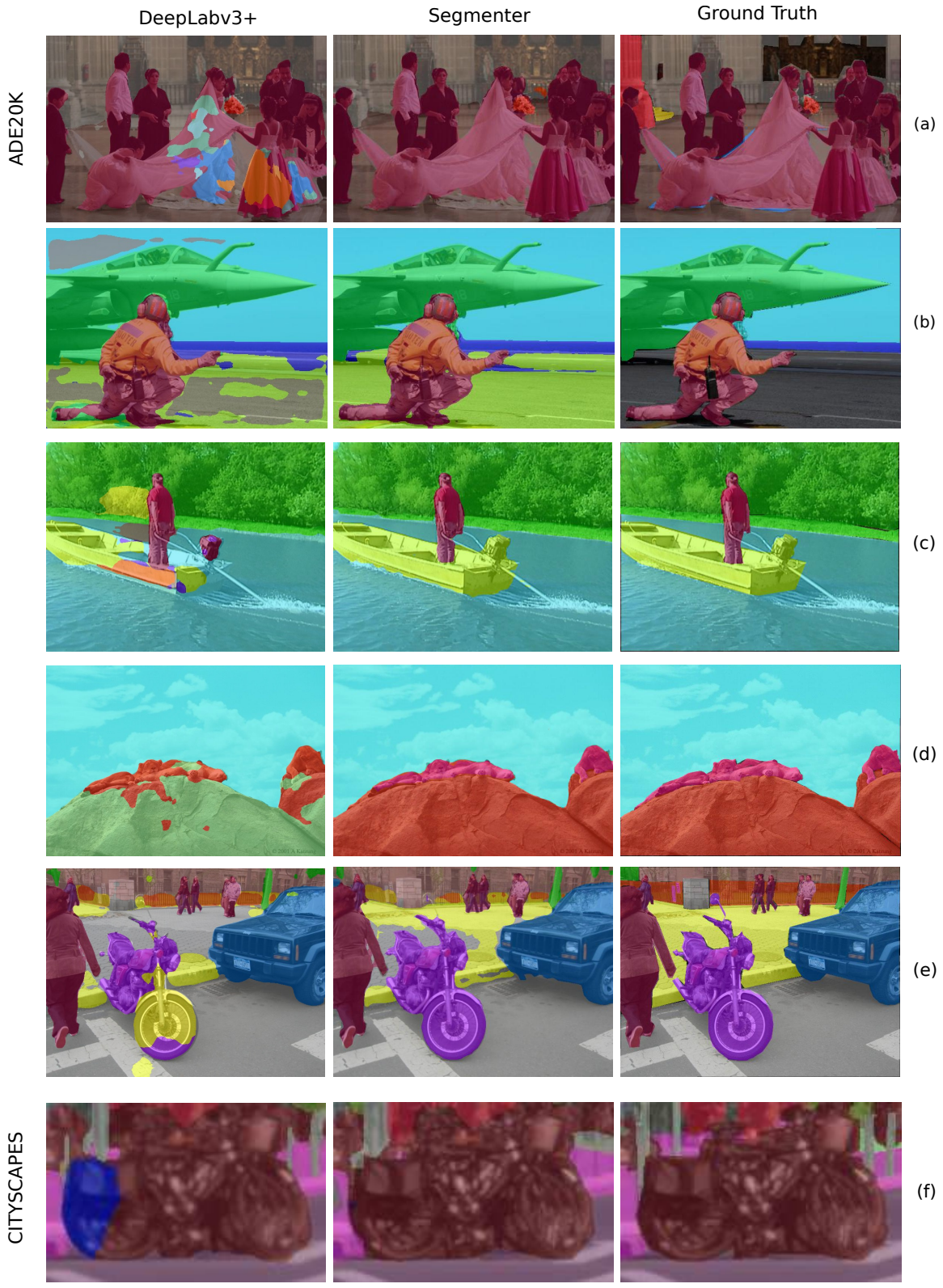


Figure 9: Segmentation maps where Seg-L-Mask/16 produces more coherent segmentation maps than DeepLabv3+ ResNeSt-101.

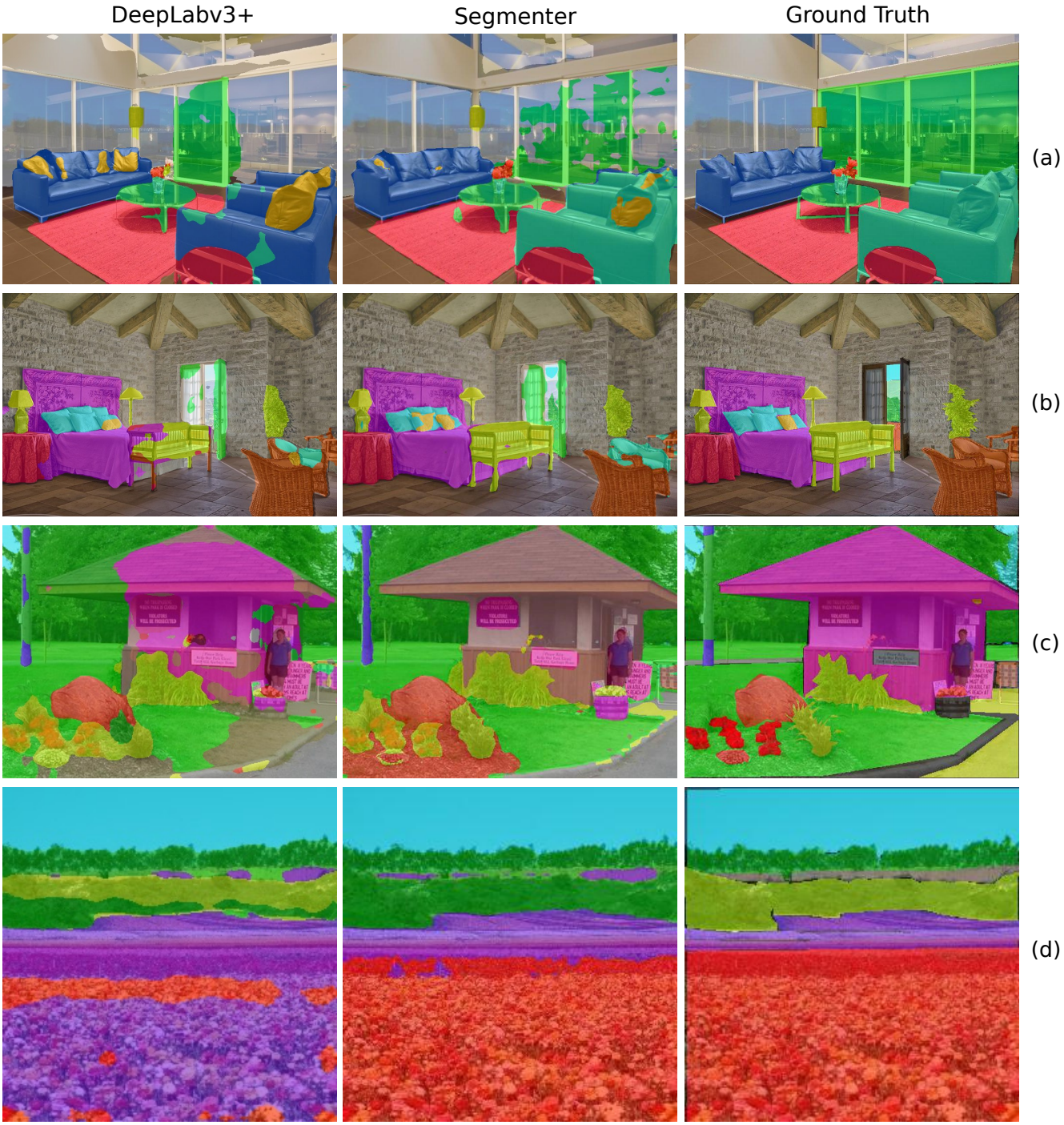


Figure 10: Examples for Seg-L-Mask/16 and DeepLabv3+ ResNeSt-101 on ADE20K, where elements which look very similar are confused.

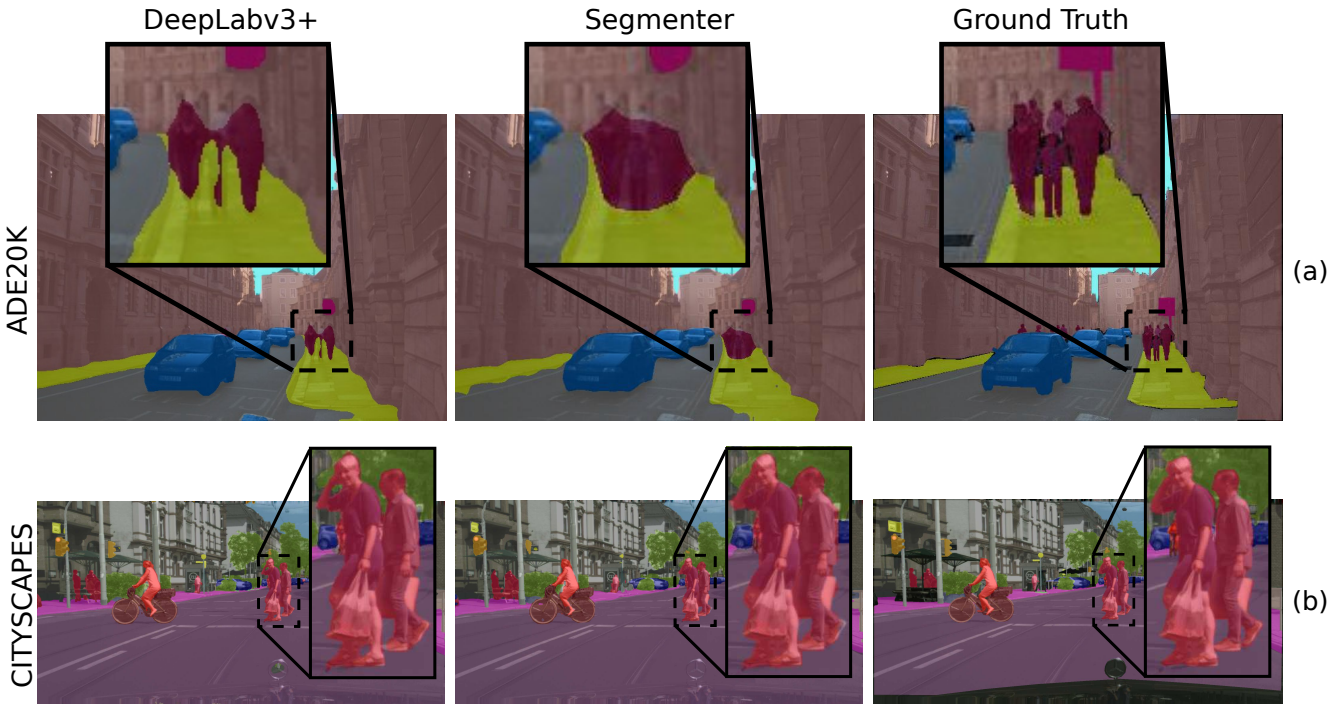


Figure 11: Comparison of Seg-L-Mask/16 with DeepLabV3+ ResNeSt-101 for images with near-by persons. We can observe that DeepLabV3+ localizes boundaries better.

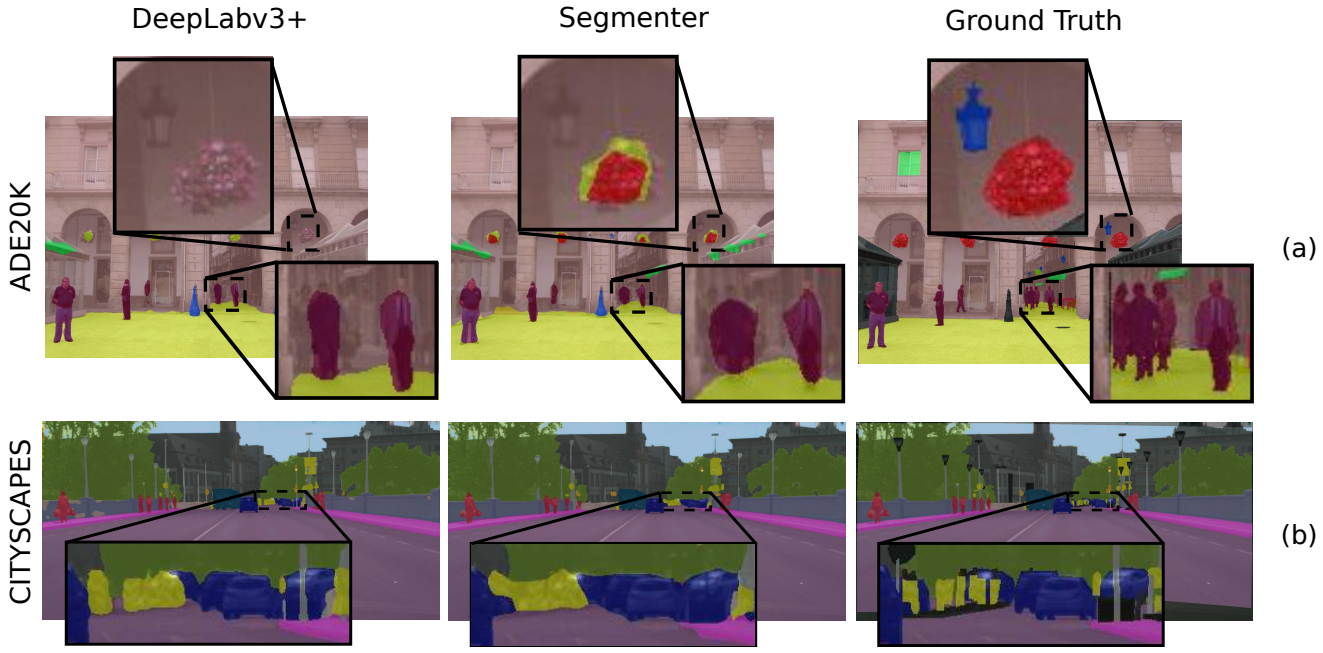


Figure 12: Failure cases of DeepLabV3+ ResNeSt-101 and Seg-L-Mask/16, for small instances such as (a) lamp, people, flowers and (b) cars, signals.