Semantic Segmentation by Early Region Proxy

Yifan Zhang   Bo Pang   Cewu Lu*
Shanghai Jiao Tong University
{zhangyf_sjtu, pangbo, lucewu}@sjtu.edu.cn

Abstract

*Typical vision backbones manipulate structured features. As a compromise, semantic segmentation has long been modeled as per-point prediction on dense regular grids. In this work, we present a novel and efficient modeling that starts from interpreting the image as a tessellation of learnable regions, each of which has flexible geometrics and carries homogeneous semantics. To model region-wise context, we exploit Transformer to encode regions in a sequence-to-sequence manner by applying multi-layer self-attention on the region embeddings, which serve as **proxies** of specific regions. Semantic segmentation is now carried out as per-region prediction on top of the encoded region embeddings using a single linear classifier, where a decoder is no longer needed. The proposed RegProxy model discards the common Cartesian feature layout and operates purely at region level. Hence, it exhibits the most competitive performance-efficiency trade-off compared with the conventional dense prediction methods. For example, on ADE20K, the small-sized RegProxy-S/16 outperforms the best CNN model using 25% parameters and 4% computation, while the largest RegProxy-L/16 achieves 52.9 mIoU which outperforms the state-of-the-art by 2.1% with fewer resources. Codes and models are available at* `https://github.com/YiF-Zhang/RegionProxy`*.*

1. Introduction

Semantic segmentation is one of the fundamental tasks in computer vision, and has been carried out using CNNs since the beginning of the deep learning era [10,17,21,27]. However, CNN is not the out-of-the-box solution for semantic segmentation considering two of its natures: 1) *Limited context.* CNN lacks of abilities to capture long range dependencies for context modeling, which is essential for semantic segmentation. 2) *Coarse prediction.* Due to its hierarchical nature, CNN outputs coarse feature which is inadequate for dense labeling. Fundamentally, the majority of semantic
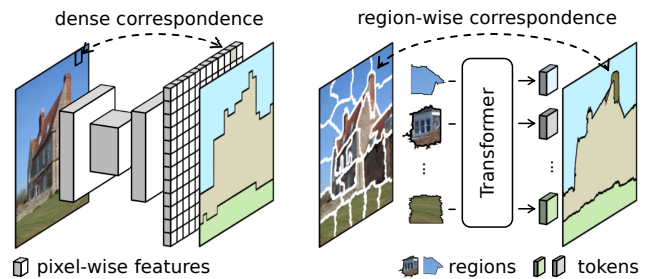
Figure 1. **Illustration of two different schemes for semantic segmentation. (Left)** Conventional encoder-decoder models establish dense correspondence between input and output on structured pixel-level grid, and segment image in a per-pixel prediction fashion. **(Right)** We propose to interpret the image as a tessellation of *learnable* regions and represent it by region-level embeddings (i.e., *proxies*) at an early stage, and segment image by *per-region prediction* using sequence-to-sequence Transformer [11,41].

segmentation researches since FCN [27] have been centring on resolving these two issues. A great number of works have been proposed for better context modeling [19,50–52] and fine-grained feature prediction [6,34,43], which significantly advance the semantic segmentation research.

Currently, the Transformer architecture [41] from natural language processing is introduced to the the vision community and has gained significant research interest. The Vision Transformer (ViT) [11] partitions image into square patches and encodes their embeddings (i.e., *tokens*) in a sequence-to-sequence manner using stacked self-attention layers. Some of its variants [26, 44] adopt a hierarchical structure and restrict self-attention in local area for better scalability. Recently, several semantic segmentation works [38, 48, 53] adopt vision Transformers as backbone and achieve impressive performances. They learn better context with the help of the inherent advantages of vision Transformers, namely the attention mechanism. However, in these models, the vision Transformer serves transparently as feature extractor which extracts 2D coarse features exactly as its convolutional counterpart does, while its main character, a sequence-to-sequence encoder, is not touched.

We revisit the aforementioned two issues: limited context and coarse prediction. While the former is a corollary

of the local receptive field which can be alleviated by adopting Transformer architecture or previous CNN-based context modules, the latter is substantially induced by the inflexible *regular* (Cartesian) layout of network features, as it does not follow the structure of real world semantics: with the large strides of typical vision backbones, it carries jumbly semantics in grid cells and brings difficulties for dense labeling. Hence, a "decoder" is required to produce fine-grained features. These facts imply that regular grids may not be the optimal feature arrangement for semantic segmentation.

In this work, we explore a novel modeling of semantic segmentation which we believe to be closer to its essence: we attempt to interpret image as a set of interrelated *regions*, where the *region* indicates a group of adjacent pixels with homogeneous semantics. As illustrated in Figure 1, we propose a simple **RegProxy** approach which *learns* regions at an early stage, explicitly models inter-region relations using Transformer [41], and encodes regions in a sequence-to-sequence fashion. We design a novel mechanism to describe region geometrics and ensure the tessellation of the entire region set, which enables us to conduct semantic segmentation by *per-region* prediction. The entire process is fully parameterized and differentiable which can be trained end-to-end efficiently. Here we present the major novelties and contributions of this work: 1) Instead of manipulating features on *regular grids*, we operate on a set of region embeddings throughout the entire network. Each of the region embeddings serves as the feature representation of a specific learnable region, namely the *region proxy*. 2) Instead of using Transformer to extract structured feature, we dive into its essence as a sequence-to-sequence encoder, and use it to explicitly model inter-region relations. 3) Instead of modeling semantic segmentation as per-pixel prediction using decoder, we segment images by directly predicting on the region embeddings using a linear classifier.

We build our model on bare ViTs [11] for image classification by adding negligible overhead (~0.5% parameters and GFLOPs), and consistently achieves the state-of-the-art performances throughout different model sizes. Extensive experiments show the competitive performance-efficiency trade-off of RegProxy under various model capacities on multiple datasets. One may peek the results in Figure 2.

2. Related Work

Vision Transformer Transformer [41] was firstly introduced in machine translation and is currently the de-facto standard of most natural language processing (NLP) tasks. The proposed attention mechanism has also inspired a number of works in computer vision [18, 45, 46]. Notably, the attention works particularly well in semantic segmentation [13, 19, 22, 55] where contextual information is crucial.

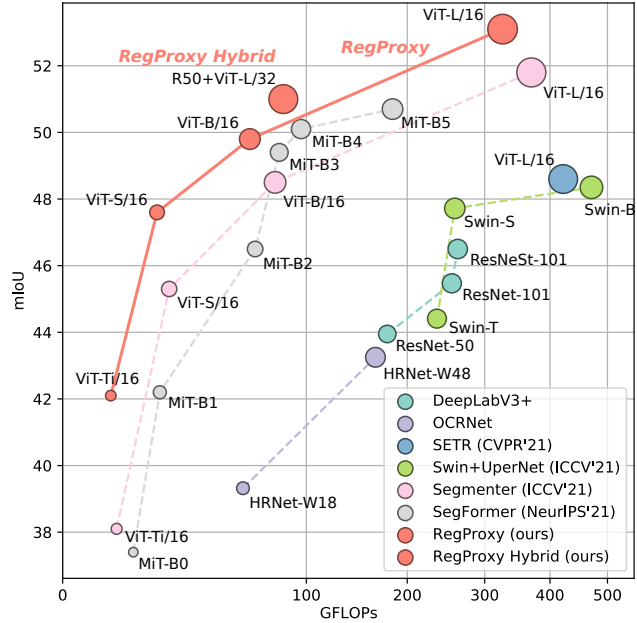Recently, Dosovitskiy et al. [11] proposes the Vi-

Figure 2. **Performance *vs.* GFLOPs on ADE20K `val` split.** We report results without multi-scale inference. RegProxy shows a highly competitive performance-efficiency trade-off among the *state-of-the-art* methods. Best viewed in color.

sion Transformer (ViT) which directly inherits the Transformer [41] architecture from NLP and works as a standalone model. ViT is gaining significant research interest and a number of improved models have been proposed. Generally, the architecture of vision Transformers can be sequential or hierarchical. Sequential models (including the original ViT [11]) partition image into patches and encode inputs in a sequence-to-sequence fashion by computing the global self-attention. DeiT [39] successfully trains ViTs on ImageNet-1k [10] with the help of strong data augmentations and knowledge distillation. CaiT [40] proposes Layer-Scale technique and later class tokens for effective training of vision Transformers at depth. XCiT [12] proposes cross-covariance attention that operates on feature dimension to build more scalable vision Transformers. Hierarchical models [26, 44] borrow some of the image-specific inductive bias from CNNs, such as translation equivariance and 2D neighborhood structure on regular grid. They compute attentions within local windows and produce hierarchical 2D features. Swin Transformer [26] is one of the most successful hierarchical vision Transformers. With the shifted window design, it brings better efficiency while allowing cross-window connections for better feature extraction.

Semantic Segmentation Semantic segmentation has been modeled as dense prediction since the emergence of fully conventional architectures. FCN [27] lays the foundation of modern semantic segmentation models, which is the first to adopt fully convolutional neural networks to segment images with arbitrary scales in an end-to-end manner. The

backbone	method	#params.	FLOPs	ADE20K	Cityscapes
ViT-Ti/16	Baseline	5.7M	3.8G	39.0 / 37.8	72.3 / 68.1
	Ours	-	-	**40.9** (+1.9)	**74.1** (+1.8)
ViT-S/16	Baseline	22.0M	14.9G	45.4 / 44.2	**76.1** / 71.8
	Ours	-	-	**46.0** (+0.6)	75.9 (−0.2)
ViT-B/16	Baseline	86.6M	58.8G	47.1 / 45.6	**78.5** / 75.1
	Ours	-	-	**47.3** (+0.2)	77.3 (−1.2)

Table 1. **Pilot experiment results.** We report baseline results with/without the bilinear upsampling of the final logits map. The baseline uses the original patch embedding [11] and works as a typical segmentation model with stride 16, while our prototype model embeds non-parametric regions (i.e., superpixels).

following works inherit the fully convolutional design and focus on better context modeling. The efforts have been put on one or more aspects that significantly improve semantic segmentation performance: 1) Enlarge receptive field using larger kernels or dilated convolution [3, 4, 31, 49]; 2) Integrate multi-scale features [5, 6, 20, 52]; 3) Refine contextual information [24, 50, 51]; 4) Utilize attention [13, 19, 22, 55]; 5) Search or design designated backbones [23, 30, 32, 43].

Recently, several works exploit Transformer to conduct semantic segmentation. SETR [53] is the first to introduce vision Transformer backbones into semantic segmentation. Segmenter [38] exploits Transformer to predict per-class masks. SegFormer [48] and DPT [32] propose hierarchical vision Transformer backbones specifically designed for dense prediction tasks. MaskFormer [7] uses the Transformer Decoder [41] to query classes with their masks from a conventional encoder-decoder model. We emphasize that our work has significant difference with these approaches: we use *bare* Transformer to model *inter-region* relations instead of using it as replaceable or extra modules of existing dense-prediction architectures.

3. Pilot Study of Region Proxy

Before introducing our main model, we first present a naive instantiation of region proxy for a proof of concept. We revisit the classical superpixel segmentation [33]. Superpixels are an over-segmentation of images that is formed by grouping perceptually similar pixels together based on low-level information (e.g., colors). It provides a low/mid-level image representation for high-level vision tasks including semantic segmentation [14, 15, 36]. In our pilot study, we adopt widely used SLIC [1] method to generate non-parametric regions to validate our region proxy notion.

Regions as Superpixels Essentially, we attempt to batch the pixel labeling by classifying superpixels. This implies that nearby pixels with similar low-level properties should be semantically homogeneous. This assumption will finally be analyzed in Section 5.3 along with our main model introduced in Section 4. At the moment, we build our prototype model under this premise.

We modify the vanilla ViT [11] which encodes patches, making it feasible to encode superpixels: N superpixels $\{\mathbf{s}_i\}$ are generated from input image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$. The irregular superpixels are cropped by their enclosing bounding boxes and resized to fixed size patches $\{\mathbf{x}_i\}$ where $\mathbf{x}_i \in \mathbb{R}^{P \times P \times 3}$, which is implemented using ROIAlign [16]. We embed $\mathbf{x}_i$ using the exact linear patch embedding in the vanilla ViT, producing N tokens with dimension D, which are then encoded in a sequence-to-sequence manner. Instead of per-pixel prediction, We directly classify regions by applying a linear classifier on every tokens. For supervision, a soft label $\hat{\mathbf{y}}_i$ is applied on the i-th token. We have

$$\hat{\mathbf{y}}_i[c] = \frac{|\{\mathbf{p} \in \mathbf{s}_i \mid \hat{y}(\mathbf{p}) = c\}|}{|\mathbf{s}_i|}, c \in \{0, 1, \ldots, K-1\} \quad (1)$$

where $\hat{y}(\mathbf{p})$ is the label of the pixel $\mathbf{p}$, and K is the number of classes. Namely, $\hat{\mathbf{y}}_i$ is the category histogram of pixels inside superpixel $\mathbf{s}_i$.

Experiment To set up the baseline, we apply the same linear classifier to the output tokens of the vanilla ViT. Hence the only difference between the baseline and our prototype model is the computation primitive: whether to be a square patch or a region. The training protocol is identical to our main experiments which is described in Section 5. In Table 1, We report results of three model variants ViT-{Ti/16, S/16, B/16} on ADE20K and Cityscapes. We observe evidential performance gains on smaller models, which vanish as the model getting larger. We hypothesize that the introduced low-level priori benefits small models, however, its intrinsic noise (i.e., inaccurate superpixels) becomes the bottleneck for larger models. We also notice the importance of the final bilinear interpolation as a standard practice: without it, the baseline model conducts patch classification which is a degeneration of the superpixel model, and yields worse performance.

4. Method

The aforementioned pilot experiments motivate us to explore for better instantiation of the region proxy modeling. In this section, we introduce the RegProxy model which computes on region embeddings (as tokens) that serve as computational proxies of specific *learnable* regions, and exploit Transformer [11, 41] to model region-wise context.

We first provide basic information about the Transformer for region context modeling in Section 4.1. We introduce the learning of region geometrics and the embedding of region features in Section 4.2. Finally, we review RegProxy model from the system perspective and describe the training/inference as per-region prediction in Section 4.3.

4.1. Transformer as Region Encoder

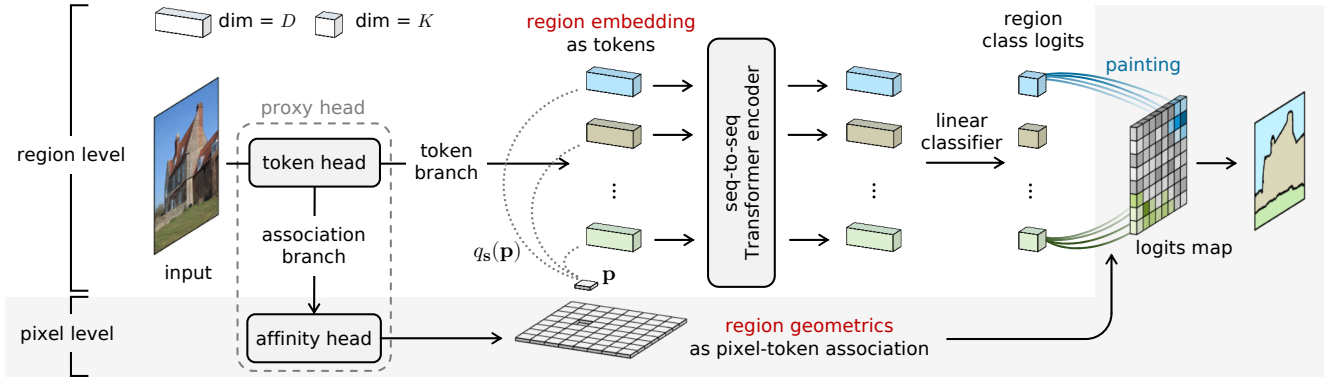Transformer is a type of sequence-to-sequence model, which applies multi-layer self-attention on its computation

Figure 3. **Overview of our RegProxy approach.** The sequence-to-sequence Transformer encoder computes on region embeddings in the form of tokens, which serve as *proxies* of specific regions whose geometrics are described by the class-agnostic pixel-token association. Notably, we model global context completely at region-level without any typical "feature map". The region embedding and its geometrical description are jointly learned using the *proxy head*. A single linear classifier is adopted for *per-region* prediction. The region class logits are simply "painted" to the output plane according the corresponding region geometrics to yield final segmentation result.
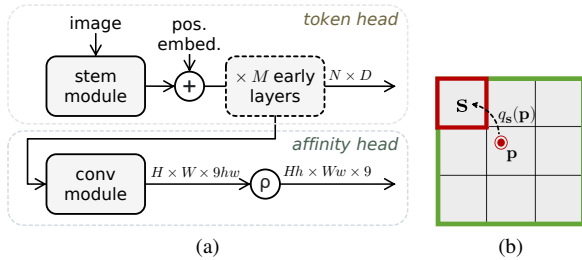

Figure 4. **More details of the RegProxy model.** (a) Illustration of the proxy head, where + stands for element-wise addition and ρ stands for reshape and rearrange of dimensions. (b) Describing region geometrics by local pixel-region association.

primitives, i.e. *tokens*. In this work, we take full advantage of the Transformer architecture to learn global context for image by directly modeling inter-region relations using self-attention. While the region learning and embedding will be discussed in Section 4.2, in this section, we briefly introduce the principle of the Transformer encoder.

Supposing N region embeddings has been produced, we write the Transformer input as a sequence of tokens:

$$\mathbf{X}_0 = \left[\mathbf{x}_{\texttt{cls}}^0, \mathbf{x}_0^0, \mathbf{x}_1^0, \ldots, \mathbf{x}_{N-1}^0\right]^\top \in \mathbb{R}^{(N+1)\times D}, \quad (2)$$

where D is the embedding dimension and the class token $\mathbf{x}_{\texttt{cls}}$ serves as a whole-image representation in image recognition tasks. A Transformer encoder is composed of L stacked layers, each of which consists of a multi-head self-attention (MSA) block followed by a multi-layer perceptron (MLP) block with two linear projection layers. Layer normalization [2] and residual connections are applied for both MSA and MLP blocks. Mathematically, we can write the computation of layer l as

$$\begin{aligned} \mathbf{A}_l &= \text{MSA}\left(\text{LN}\left(\mathbf{X}_{l-1}\right)\right) + \mathbf{X}_{l-1}, \\ \mathbf{Z}_l &= \text{MLP}\left(\text{LN}\left(\mathbf{A}_{l-1}\right)\right) + \mathbf{A}_{l-1}, \end{aligned} \quad (3)$$

where the output $\mathbf{Z}_l$ is used as input of the next layer.

The Vision Transformer (ViT) inherits the exact architec-ture from the Transformer in NLP [41], accompanied with a stem module designed for vision tasks. In this work, we follow the standard definition of the ViT models in [11, 39]. Our RegProxy-{Ti/16, S/16, B/16, L/16} models are based on vanilla ViTs which use patch embedding stems, while our RegProxy-{R26+S/32, R50+L/32} models are based on hybrid ViTs which use CNN stems. These ViTs vary from total strides and numbers of layers, embed dimensions and attention heads. The detailed specifications can be found in Table 2 and are described in Section 5.

4.2. Learning Region Proxy

As depicted in Figure 3, our *region proxy* method comes with two important components: the region feature and its geometrics. In this section, we introduce the region feature embedding and its geometrical description.

Describing Region Geometrics   Learning and describing region geometrics is not quite straightforward due to the irregular shape and various scales. An intuitive idea is to predict a binary mask for each region, either to be image-sized or within a local area. However, we find neither of the approaches practical for our sequence-to-sequence modeling. Predicting a full-sized mask for each token will be computationally expensive, while predicting small sized local masks cannot ensure the tessellation of regions, consequently causing overlapping and non-predicted areas.

To this end, we propose a novel mechanism to describe region geometrics by pixel-to-token association. We start from an initial $H \times W$ grid where $H \times W = N$. Each token lays on a single cell which serves as a "seed" of its corresponding region $\mathbf{s}$. Note the cell itself is merely a token location indicator and has nothing to do with the actual region geometrics. We build the pixel-to-token association by assigning each pixel $\mathbf{p} = (u, v)$ to region $\mathbf{s}$ with a probability $q_{\mathbf{s}}(\mathbf{p})$. It is unnecessary to apply this association globally. Instead, we associate pixel $\mathbf{p}$ only with tokens that lays

in its neighborhood $\mathcal{N}_{\mathbf{p}}$ which satisfies

$$\sum_{\mathbf{s} \in \mathcal{N}_{\mathbf{p}}} q_{\mathbf{s}}(\mathbf{p}) = 1 \qquad (4)$$

This is illustrated in Figure 4b, as the pixel $\mathbf{p}$ is being assigned to one of the 9 region seeds (marked with red box) with in its surrounding neighborhood (in green box). Mathematically, we can write the association as a class-agnostic affinity map $\mathbf{Q} \in \mathbb{R}^{(Hh) \times (Ww) \times |\mathcal{N}_p|}$ at pixel-level which describe the actual geometries of all regions, where $(Hh) \times (Ww)$ matches the size of the output segmentation map and (h, w) is the relative stride of the initial token grid. Empirically, we set the neighborhood size to 3×3 and $|\mathcal{N}_p| = 9$, which works well through-out all model sizes.

For an intuitive interpretation, the region is represented by a "core" (i.e., token's location on the initial $H \times W$ cell) and pixels surround it by probabilities. With the constraint of Eq. 4, the entire set of probabilistic regions forms a tessellation that covers the image plane with no overlaps and no gaps. Meanwhile, the regions learn to be highly semantically homogeneous even without explicit regularization, which will be discussed in Section 5.3. These facts enable us to segment image in a per-region prediction fashion.

Embedding Regions We jointly embed region features and learn their geometrical description $\mathbf{Q}$ using the *proxy head* depicted in Figure 4a. To embed region features into tokens, we use a small part of the ViT backbone as *token head*. We take the exact stem module in [11] along with learnable position embeddings, which produces N tokens of dimension D where $N = H \times W$. The stem module can be linear patch projection for vanilla ViT models, or CNNs for hybrid models (see Section 5). For our vanilla ViT models, we further apply the first M Transformer layers for better feature extraction.

To predict region geometrical description $\mathbf{Q}$, we design a lightweight *affinity head* that is highly integrated with the ViT backbone, bringing negligible extra parameters and computations. As illustrated in Figure 4a, we predict the affinity map $\mathbf{Q}$ directly using features from the token head. Specifically, the token feature $\mathbf{T} \in \mathbb{R}^{N \times D}$ is reshaped to $\mathbf{T}' \in \mathbb{R}^{H \times W \times D}$ according to its initial layout, and then fed into a convolution module. The convolution module fuses local region-wise information and produces $H \times W \times (9hw)$ feature map, as we predict affinity vectors for all hw pixels inside each cell in batches. The compact map is then "unpacked" to the pixel-level format, which finally yields the affinity map $\mathbf{Q} \in \mathbb{R}^{(Hh) \times (Ww) \times 9}$.

In details, the convolution module consists of one 3×3 depth-wise conv layer, followed by a 1×1 conv layer. The affinity map is activated by a Softmax to produce normalized probabilities. Notably, we discover some interesting facts about choices of the number of early layers M for affinity prediction, which will be discussed in Section 5.3.

4.3. Segmentation by Per-Region Prediction

By our assumption, region embedding carries homogeneous semantics throughout the Transformer network. This enables us to directly predict on regions instead of pixels as opposed to conventional FCN-style models [6, 26, 27, 48], hence significantly reduces the number of predictions. As depicted in Figure 3, we apply a **linear** classifier on the last layer output $\mathbf{X}_L$ of the Transformer backbone, which we find sufficient to yield strong performance. The linear classifier produces class logits for all tokens except $\mathbf{x}_{\texttt{cls}}$:

$$\mathbf{Y} = \left[\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{N-1} \right]^{\top} \in \mathbb{R}^{N \times K}, \qquad (5)$$

where K is number of classes.

We can easily attain the pixel-level segmentation result using the region geometrical description $\mathbf{Q}$ introduced in Section 4.2. We simply "paint" the token logits to the corresponding regions as illustrated in Figure 3. In practice, we paint in a per-pixel manner for efficient implementation. For pixel $\mathbf{p} = (u, v)$, its class logits is calculated by

$$\mathbf{Y}'[u, v] = \sum_{\mathbf{s} \in \mathcal{N}_{\mathbf{p}}} \mathbf{y}(\mathbf{s}) \cdot q_{\mathbf{s}}(\mathbf{p}), \qquad (6)$$

where $\mathbf{Y}' \in \mathbb{R}^{(Hh) \times (Ww) \times K}$ is the output logits map, $\mathbf{y}(\mathbf{s})$ is the class logits of the token corresponding to region $\mathbf{s}$. In practice, we set $(h, w) = (4, 4)$, yielding $\times 4$ stride logits map for RegProxy-x/16 models, and $\times 8$ stride logits map for RegProxy-x/32 models. Our model is trained end-to-end using cross-entropy loss *without* class balance or hard example mining. The output logits map is upsampled to ground-truth/input size for supervision during training and prediction during inference following the standard practice.

5. Experiments

Datasets and Metrics We study RegProxy on three public datasets: ADE20K [54], Cityscapes [9] and Pascal Context [29]. ADE20K is a challenging scene parsing dataset with 150 classes, which contains $20\,210$ images for training and 2000 images for validation. Cityscapes is a high-resolution street scene dataset with 19 classes. We train on the 5000 fine annotated training images and test on 500 validation images. Pascal Context contains 4996 training images and 5104 validation images with 60 classes (including one background class). We report Intersection over Union averaged over all classes (mIoU).

ViT Backbones We build our model on the original ViTs as defined in [11,37]. We conduct extensive experiments on six variants: vanilla ViTs with four specs (tiny, small, base and large) and hybrid ViTs with two specs (small, large). Vanilla ViTs use *patch embedding* as stem which partitions an image into $P \times P$ patches, while each patch is flattened and linearly projected to an embedding with D dimension. For hybrid ViTs, the projection is applied to the features ex-

backbone	stride	stem	#layers	dim	#heads	#params
Ti/16	16	patch embed.	12	192	3	6M
S/16	16	patch embed.	12	384	6	22M
B/16	16	patch embed.	12	768	12	86M
L/16	16	patch embed.	24	1024	16	307M
R26+S/32	32	ResNet[2, 2, 2, 2]	12	384	6	37M
R50+L/32	32	ResNet[3, 4, 6, 3]	24	1024	16	330M

Table 2. **Configurations of ViT variants.** (Top) Vanilla ViTs use patch embedding. (Bottom) Hybrid ViTs use ResNet [17] stem.

tracted by a residual convolutional network [17], hence the patch size is equivalent to CNN's stride S. We set $P = 16$ for vanilla ViTs and $S = 32$ for hybrid ones. Namely, the initial $H \times W$ token grid is of strides 16 and 32 respectively. Larger stride generally impairs performance while requiring less computation. The detailed configurations can be found in Table 2. We use weights pre-trained on ImageNet-21k [37] following recent works [26, 38, 53].

Implementation Details We implement our method using the public codebase mmsegmentation [8]. We introduce minimum changes to its default settings that are widely used by the community. For training, we use input sizes of 512×512, 768×768 and 480×480 for ADE20K, Cityscapes and Pascal Context, respectively. We train our "Large" model on ADE20K using a 640×640 crop following [26, 38, 48]. We use AdamW [28] optimizer with an initial learning rate of 6×10^{-5}, a weight decay of 10^{-2} and a "poly" learning rate scheduler [4] with $power = 1.0$ following [12, 26, 48]. We use a batch size of 16 and train for 160k, 80k, 40k iterations on ADE20K, Cityscapes and Pascal Context, respectively. We keep data augmentations and all other training settings identical to default settings in [8]. Training tricks such as hard example mining, auxiliary losses or class balance loss are *not* included.

For testing, we use the sliding window mode with window sizes matching the crop sizes for training. We use default window strides in [8]. We report both single scale results and multi-scale + flipping results with scaling factors of $\{0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$.

5.1. Main Results

Baselines To setup the baseline, we simply append a linear classifier to the bare ViTs to produce per-patch prediction $\mathbf{Y} \in \mathbb{R}^{N \times K}$. Then the patch-wise logits are reshaped to their original 2D layout $\mathbf{Y}' \in \mathbb{R}^{H \times W \times K}$ and upsampled to image size for training and inference following the standard segmentation pipeline. It can be interpreted as a typical segmentation model of stride 16. This setting is identical to the "Linear" baseline in [38], and we produce similar results by our implementation. Since they have no structural difference with the vanilla ViTs [11] for image classification, we simply refer to the baseline models as **ViT-x/16** where $x \in \{\text{Ti, S, M, L}\}$.

method	FLOPs	#params.	ADE20K (SS/MS)		Cityscapes (SS/MS)	
ViT-Ti/16	3.8G	5.7M	39.0	39.8	72.3	74.1
+Mask.T	+1.0G	+1.0M	38.1 (−0.9)	38.8 (−1.0)	-	-
+Ours	+0.1G	+0.1M	**42.1** (+3.1)	**43.1** (+3.3)	**76.5** (+4.2)	**77.7** (+3.6)
ViT-S/16	14.9G	22.0M	45.4	45.9	76.1	78.0
+Mask.T	+4.2G	+4.1M	45.3 (−0.1)	46.9 (+1.0)	-	-
+Ours	+0.2G	+0.2M	**47.6** (+2.2)	**48.4** (+2.5)	**79.8** (+3.7)	**81.5** (+3.5)
ViT-B/16	58.8G	86.6M	47.1	48.1	78.5	80.5
+UperNet	+336G	+57.6M	47.9 (+0.8)	49.5 (+1.4)	79.6 (+1.1)	80.9 (+0.4)
+Mask.T	+17.1G	+16.0M	48.7 (+1.6)	50.1 (+2.0)	-	80.6 (+0.1)
+Ours	+0.7G	+0.7M	**49.8** (+2.7)	**50.5** (+2.4)	**80.9** (+2.4)	**82.2** (+1.7)
ViT-L/16	325.0G	304.3M	50.7	51.8	78.4	80.7
+Mask.T	+44.5G	+28.5M	51.8 (+1.1)	53.6 (+1.8)	-	81.3 (+0.6)
+Ours	+0.9G	+1.8M	**52.9** (+2.2)	53.4 (+1.6)	**81.4** (+3.0)	**82.7** (+2.0)
SETR [53]	325.1G	305.6M	48.1	48.8	77.9	-
+MLA	+8.7G	+4.0M	48.6 (+0.5)	50.3 (+1.5)	77.2 (−0.7)	-
+PUP	+97.5G	+11.7M	48.6 (+0.5)	50.1 (+1.3)	79.3 (+1.4)	-

* In green are the gaps of at least **+2.0** mIoU.

Table 3. **Compare different integrations of vision Transformer.** We report the results of the baseline, the state-of-the-art Mask Transformer from Segmenter [38], UperNet [47] and our RegProxy. We also report numbers from SETR [53] for reference.

Results We compare different integrations of vision Transformer for semantic segmentation. Table 3 summarizes the results. Our RegProxy approach brings consistent performance gains throughout all model sizes with an evidential gap (2~3 mIoU on ADE20K and 2~4 mIoU on Cityscapes), while costing negligible overhead (less than 2 M parameters and 1 GFLOPs computation for the largest model). It also significantly outperforms the Segmenter [38] which carries a heavy decoder. Moreover, we notice the performance degeneration of Segmenter on smaller models, which is not observed on our approach. As a reference, we also report results from SETR [53] which appends different convolutional decoders to the ViT-L/16 backbone and the UperNet [47] results ran by us. By these results, we may suggest that our region proxy modeling is a more effective and efficient way to exploit vision Transformers in semantic segmentation compared with the encoder-decoder scheme.

5.2. Comparison across Architectures

ADE20K We conduct comprehensive comparisons with the state-of-the-art methods on ADE20K. Results in Figure 2 show that the RegProxy is among the most competitive models in terms of performance-efficiency trade-off. RegProxy consistently outperforms the state-of-the-art Segmenter [38] by a large margin, which uses the same ViT backbones and pre-training as ours. It also shows significant superiority compared with recent SegFormer [38], Swin-Transformer [26] and SETR [53]. Table 4 gives a more detailed comparison with respect to parameters, GFLOPs, inference speed and performance. The smallest RegProxy-Ti/16 achieves 42.1 mIoU with only 5.8 M parameters and 3.9 GFLOPs, and runs at a speed of 38.9 FPS, which out-
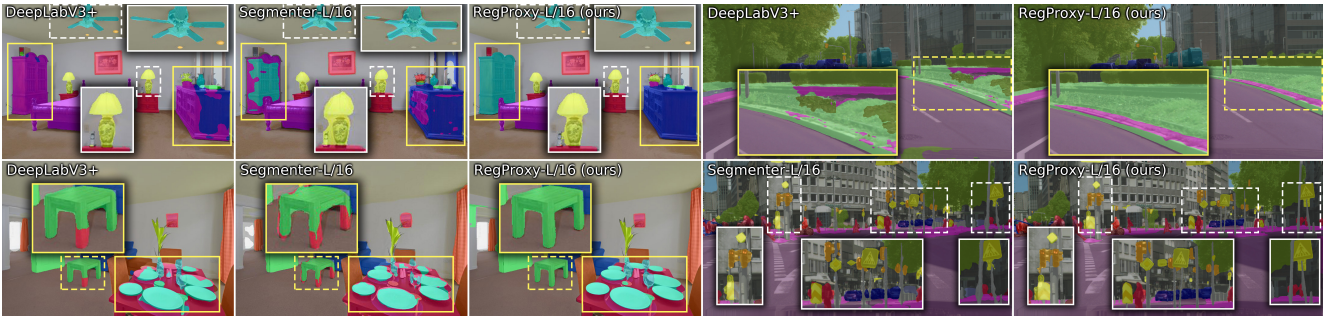
Figure 5. **Qualitative comparison on ADE20K and Cityscapes.** Our model produces much more accurate contour (in white box) compared to Segmenter [38] and captures more consistent context (in yellow box) compared to DeepLabV3+ [6]. Best viewed zoom in.

method	backbone	FLOPs	#params.	mIoU	FPS
FCN [27]	MobileNetV2	39.6G	9.8M	19.7 / -	64.4
D.LabV3+ [6]	MobileNetV2	69.4G	15.4M	34.0 / -	43.1
SegFormer [48]	MiT-B0	8.4G	3.8M	37.4 / 38.0	50.5
Segmenter [38]	ViT-Ti/16	4.9G	6.7M	38.1 / 38.8	-
RegProxy	ViT-Ti/16	3.9G	5.8M	**42.1** / **43.1**	38.9
OCRNet [50]	HRNet-W18	55G	12M	39.3 / 40.8	18.9
SegFormer [48]	MiT-B1	16G	14M	42.2 / 43.1	47.7
Segmenter [38]	ViT-S/16	19G	26M	45.3 / 46.9	29.8
MaskFormer [7]	Swin-T	55G	42M	46.7 / 48.8	22.1
RegProxy	ViT-S/16	15G	22M	47.6 / 48.5	32.1
RegProxy	R26+ViT-S/32	16G	36M	47.8 / 49.1	28.5
OCRNet [50]	HRNet-W48	165G	71M	43.2 / 44.9	17.0
D.LabV3+ [6]	ResNet-101	255G	63M	45.5 / 46.4	14.1
D.LabV3+ [6]	ResNeSt-200	345G	88M	- / 48.4	-
Segmenter [38]	ViT-B/16	76G	103M	48.7 / 50.1	14.6
RegProxy	ViT-B/16	59G	87M	49.8 / 50.5	20.1
DPT [32]	DPT-Hybrid	-	123M	- / 49.0	-
SETR [53]	ViT-L/16	422G	318M	48.6 / 50.1	4.5
SegFormer [48]	MiT-B5	183G	85M	51.0 / 51.8	9.8
Segmenter [38]	ViT-L/16	370G	333M	51.8 / **53.6**	-
RegProxy	R50+ViT-L/32	82G	323M	51.0 / 51.7	12.7
RegProxy	ViT-L/16	326G	306M	**52.9** / 53.4	6.6

* All models in the last group except DPT use a larger 640×640 crop.

Table 4. **Comparison to state-of-the-art methods on ADE20K val split.** We report both single/multi-scale results. We group the methods based on model capacities and computational costs for fine-grained comparison. Cross group comparison is welcomed.

method	backbone	crop	FLOPs	#params.	mIoU
D.LabV3+ [6]	ResNet-18	769^2	992G	12M	76.3 / 77.9
OCRNet [50]	HRNet-W18	full	424G	12M	78.6 / 80.5
SegFormer [48]	MiT-B0	768^2	52G	4M	75.3 / -
SegFormer [48]	MiT-B1	1024^2	244G	14M	78.5 / 80.0
RegProxy	ViT-Ti/16	768^2	69G	6M	76.5 / 77.7
RegProxy	ViT-S/16	768^2	270G	23M	**79.8** / **81.5**
OCRNet [50]	HRNet-W48	full	1297G	70M	80.7 / 81.9
Auto-D.Lab [25]	NAS-F48	769^2	-	44M	- / 80.4
Axial-D.Lab [42]	Axial-D.Lab-XL	-	2447G	173M	- / 81.1
D.LabV3+ [6]	ResNeSt-200	full	-	88M	- / **82.7**
SETR [53]	ViT-B/16	768^2	-	98M	79.5 / -
SETR [53]	ViT-L/16	768^2	-	318M	79.3 / 82.2
Segmenter [38]	ViT-B/16	768^2	1344G	103M	- / 80.6
Segmenter [38]	ViT-L/16	768^2	-	337M	79.1 / 81.3
RegProxy	ViT-B/16	768^2	1064G	88M	81.0 / 82.2
RegProxy	ViT-L/16	768^2	-	307M	**81.4** / **82.7**

Table 5. **Comparison to state-of-the-art methods on Cityscapes val split.** The "full" crop indicates the whole image inference, while others indicate the *sliding window* protocol.

performs the best efficient models by large margins (greater than 4.0 mIoU). Our small model RegProxy-S/16 has surpassed the heaviest CNN model (DeepLabV3+ w/ ResNeSt-200) using only 25% parameters and 4% computation. The medium sized RegProxy-B/16 significantly outperforms all CNN models with much less computation. The largest RegProxy-L/16 achieves 52.9 mIoU *without* multi-scale inference, outperforms Segmenter [38] by 1.1 mIoU. The hybrid models also achieve notable results even with a large-stride backbone. The RegProxy-[R26+S/32] achieves 47.8/49.1 mIoU, and the RegProxy-[R50+L/32] achieves a comparable 51.0/51.7 mIoU among jumbo Transformer-based models using only 1/4 computations. And gener-

ously, our RegProxy runs at a significantly higher frame rate among different sized models thanks to the concise design.

Cityscapes and Pascal Context In Table 5 we compare the state-of-the-art methods on Cityscapes. We observe similar results as on ADE20K. Our RegProxy-{Ti/16, S/16} outperform their counterparts with a ~1.2 mIoU margin. Our RegProxy-{B/16, L/16} also achieve state-of-the-art performances compared to larger models. Notably, they outperform SETR [53] and Segmenter [38] by ~2 mIoU using identical backbones. We notice that on Cityscapes, the performance gained from model capacities tends to saturate on larger models. RegProxy-L/16 is only 0.4 mIoU higher than RegProxy-B/16 (the gap on ADE20K is 2.2 mIoU using both 512×512 crops). This is also observed on SETR [53] and Segmenter [38]. We hypothesize that this is due to the simpler context of Cityscapes. We report results on Pascal Context in Table 6. RegProxy-L/16 achieves 58.4 mIoU and significantly outperforms the state-of-the-art CNN models, and achieves comparable performance with Segmenter [38] using less resources. We present qualitative comparison in Figure 5 and supplemental material.

method	backbone	FLOPs	#params.	mIoU
DeepLabV3+ [6]	ResNet-101	224G	63M	47.2 / 48.3
OCRNet [50]	HRNet-W48	144G	71M	- / 56.2
SETR [53]	ViT-L/16	238G	318M	54.9 / 55.8
Segmenter [38]	ViT-B/16	67G	102M	- / 55.0
Segmenter [38]	ViT-L/16	210G	333M	58.1 / **59.0**
RegProxy	ViT-B/16	52G	87M	55.2 / 55.4
RegProxy	ViT-L/16	183G	305M	**58.4** / 58.8

Table 6. **Comparison to state-of-the-art methods on Pascal Context `val` split.**

5.3. Analysis and Ablation Study

In this section, we present the most significant analysis and ablation studies. Due to limited space, we present more of them in the supplemental material.

Region Semantics We analyze the semantical homogeneity of regions, which is described using "region entropy": we calculate the per-pixel category histogram within every regions using Eq. 1 and compute their entropies [35]. We analyze the distribution of region entropies on the entire ADE20K `val` split, with regard to three sources: regular cell used by conventional segmentation model such as [38], superpixels used in our pilot study, and learned regions from our RegProxy approach. The results are shown in Figure 6. Although we do not apply any explicit regularization, the learned regions still exhibit highly compact semantics compared to regular cells and superpixels. In the right part of Figure 6, we visualize the leaned probabilistic regions of a number of tokens on tiny crops of Cityscapes validation images. Note the regions are *class-agnostic* (as association of pixels and nearby tokens) and leaned from shallow features. However, they still capture fine-grained boundaries of high level classes. This suggest that the computation primitives in our model (i.e., tokens) carries more uniformed semantics compared to CNNs or other vision Transformer models that compute on structured features, which probably leads to easier optimization hence the better performances.

Per-Class Performance In Figure 7, we analyze per-class performance on Cityscapes. Compared to Transformer-based Segmenter [38], DeepLabV3+ [6] is good at handling small/thin classes (e.g., traffic sign, pole) thanks to its small stride, yet fails more on confusing classes (e.g., bus, wall) which require wider context. Our region proxy modeling possesses both of their merits and consistently outperforms or being comparable to both of the methods on all classes.
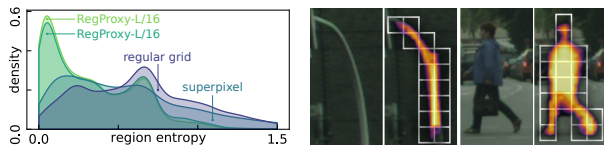

Figure 6. **(Left) Distribution of region entropies** estimated by kernel density estimation (KDE). Regions with entropy of 0 are ignored. **(Right) Geometrics of the *class-agnostic* regions** and its corresponding tokens (marked using white cell).

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	
	98.1	85.1	92.4	62.7	64.8	54.1	66.8	76.6	92.1	67.7	94.4	80.5	64.7	95.0	86.5	92.2	86.3	67.6	76.2	Seg.
	+0.1	+0.6	+0.7	-4.0	-2.0	+15.0	+7.1	+4.5	+0.6	-1.8	+1.0	+3.3	+2.7	+0.7	-7.5	-4.5	-5.6	+0.4	+2.6	D.Lab
	+0.1	+1.2	+0.8	-0.7	+0.5	+11.8	+5.8	+4.3	+0.8	+1.7	+0.9	+3.2	+3.6	+0.8	+0.6	-1.1	-0.8	+6.1	+3.4	Ours

Figure 7. **Per-class performance on Cityscapes.** We report relative mIoU of DeepLabV3+ and our **RegProxy-L/16** model compared with Segmenter-L/16 [38].

This is credited to the *early region* design that captures fine regions in advance and models region-wise relations afterwards, hence learns global context without loss of details.

Depth of Token Head As illustrated in Figure 4a, we use the first M Transformer layers in the token head. M is set to 0 for our hybrid models since the convolutional stem is strong enough for region learning and embedding. Here we investigate the M setting for vanilla ViT based models. The results are displayed in Table 7. We find that using early layers (e.g., 3~5 layers) generously works well, while setting M to 0 (a too shallow token head) will harm the performance. Interestingly and importantly, we also find that a too deep token head will cause a significant performance drop. To the extreme, setting $M = 12$ (uses all Transformer layers, the proxy head degenerates to a typical decoder) will yield a performance no better than the baseline's. This indicates the importance of the Transformer layers *after* the proxy head, which model context for learned regions.

depth M	0	3	4	5	6	9	12	baseline
ADE20K	46.3	47.1	47.0	**47.2**	46.8	46.6	45.7	45.0
Cityscapes	76.8	**79.0**	78.8	78.2	77.7	76.1	75.2	75.4

Table 7. **Ablation on depth of token features used for region learning**. We report single scale mIoU results of RegProxy-S/16 on ADE20K and Cityscapes using half of the training schedule.

6. Conclusion

In this paper, we present *region proxy*, a novel and efficient modeling of semantic segmentation. It interprets the image as a tessellation of learnable regions, each of which has flexible geometrics and carries homogeneous semantics. We conduct semantic segmentation by per-region prediction on top of region embeddings, which are encoded using Transformer in a sequence-to-sequence fashion. Without a decoder, the RegProxy segmentation models still exhibit the most competitive performance-efficiently trade-off among its dense prediction counterparts. We hope our region proxy modeling provide an inspiring perspective of efficient image representation for semantic segmentation and other vision tasks.

7. Acknowlegement

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012. 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 3

[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, 2017. 3, 6

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

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 1, 3, 5, 7, 8

[7] Bowen Cheng, Alexander G Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *arXiv preprint arXiv:2107.06278*, 2021. 3, 7

[8] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 6

[9] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 5

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3, 4, 5, 6

[12] Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021. 2, 6

[13] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 2, 3

[14] Raghudeep Gadde, Varun Jampani, Martin Kiefel, Daniel Kappler, and Peter V Gehler. Superpixel convolutional networks using bilateral inceptions. In *European conference on computer vision*, pages 597–613. Springer, 2016. 3

[15] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International journal of computer vision*, 80(3):300–316, 2008. 3

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[17] 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. 1, 6

[18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[19] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019. 1, 2, 3

[20] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 3

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[22] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9167–9176, 2019. 2, 3

[23] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8553–8562, 2020. 3

[24] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017. 3

[25] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019. 7

[26] 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 arXiv:2103.14030*, 2021. 1, 2, 5, 6

[27] 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. 1, 2, 5, 7

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[29] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014. 5

[30] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9126–9135, 2019. 3

[31] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2017. 3

[32] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021. 3, 7

[33] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Computer Vision, IEEE International Conference on*, volume 2, pages 10–10. IEEE Computer Society, 2003. 3

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[35] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 8

[36] Abhishek Sharma, Oncel Tuzel, and Ming-Yu Liu. Recursive context propagation network for semantic scene labeling. In *NIPS*, volume 1, page 2, 2014. 3

[37] 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 arXiv:2106.10270*, 2021. 5, 6

[38] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021. 1, 3, 6, 7, 8

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2, 4

[40] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 2

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2, 3, 4

[42] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Standalone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020. 7

[43] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 1, 3

[44] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 2

[45] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[46] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2

[47] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 6

[48] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021. 1, 3, 5, 6, 7

[49] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 3

[50] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 173–190. Springer, 2020. 1, 3, 7, 8

[51] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7151–7160, 2018. 1, 3

[52] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 1, 3

[53] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021. 1, 3, 6, 7, 8

[54] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 5

[55] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 593–602, 2019. 2, 3