StructToken: Rethinking Semantic Segmentation With Structural Prior

Fangjian Lin©, Zhanhao Liang©, Sitong Wu, Junjun He©, Kai Chen, and Shengwei Tian©

Abstract—In previous deep-learning-based methods, semantic segmentation has been regarded as a static or dynamic per-pixel classification task, i.e., classify each pixel representation to a specific category. However, these methods only focus on learning better pixel representations or classification kernels while ignoring the structural information of objects, which is critical to human decision-making mechanism. In this paper, we present a new paradigm for semantic segmentation, named structure-aware extraction. Specifically, it generates the segmentation results via the interactions between a set of learned structure tokens and the image feature, which aims to progressively extract the structural information of each category from the feature. Extensive experiments show that our StructToken outperforms the state-of-the-art on three widely-used benchmarks, including ADE20K, Cityscapes, and COCO-Stuff-10K.

Index Terms—Semantic segmentation, transformer.

I. INTRODUCTION

With the development of self-driving technology [16], human-computer interaction [18], and augmented reality [1], semantic segmentation has attracted more and more attention.

Since the deep-learning era, semantic segmentation has been mainly formulated as a per-pixel classification task, that is, classifying each pixel to a specific category from a learned classifier (such as a 1 × 1 convolution). According to the property of classifier, previous works can be categorized as two paradigms: static per-pixel classification and dynamic per-pixel classification. As shown in Figure 1a, for the static per-pixel classification paradigm, the classifier is fixed after the training process. The methods following this paradigm mainly focused on how to learn better representation for each pixel through context modeling [7], [28], [45], [53], [49] or automatic architecture design [11], [41], [47]. As the static classifier learned from the dataset can be regarded as a comprehensive representation of each class, which may not be consistent with the representation of each object in every image, some recent works [9], [10], [38] proposed to dynamically learn a classifier for different inputs according to their own contents. As shown in Figure 1b, the initial kernel is updated by the image feature, resulting in a dynamic classifier more adaptive to the current input.

In these two paradigms, the entire decoder is dedicated to learning better features (including precise semantics and details) and a more robust classifier, and the segmentation decision is only performed in the final segmentation head via the per-pixel classification. However, from the human perspective, the decision-making process of semantic segmentation presents a different pattern. In particular, based on the underlying knowledge of category-wise structural information (such as texture, shape and spatial layout), human beings first determine the rough area of each category and then gradually refine it, rather than paying close attention to all the image information at first and final performing the classification at one time. This motivates us to explore whether a paradigm more in line with the human decision-making process is better than the previously popular per-pixel classification for semantic segmentation.

In this paper, we design a human-like paradigm for semantic segmentation, named structure-aware extraction. To simulate human knowledge, we define a set of learnable structure tokens, each of which is expected to model the implicit structural information of one category. As shown in Figure 1c, given the image feature, the structure tokens gradually extract information from the image feature. Qualitative visualization shows that the structural information becomes more and more explicit during progressive extraction. Thus, the refined structure tokens of the final layer can be directly regarded as the segmentation result. Obviously, our paradigm is similar to the human-like process that uses structural knowledge to make rough discrimination first and then gradually perform refinement.

Following our structure-aware extraction paradigm, we further design a semantic segmentation network, named StructToken, to evaluate the effectiveness of our paradigm. As mentioned above, the extraction aims to construct the

©Fangjian Lin is with the School of Software, Xinjiang University, Urumqi 830046, China, and also with the Shanghai AI Laboratory, Shanghai 200032, China.
©Zhanhao Liang is with the Shanghai AI Laboratory, Shanghai 200032, China, and also with the School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China.
©Sitong Wu is with Baidu Research, Beijing 100094, China.
©Junjun He is with the Shanghai AI Laboratory, Shanghai 200032, China.
©Kai Chen is with the Shanghai AI Laboratory, Shanghai 200032, China, and also with SenseTime Research, Shanghai 518066, China.
©Shengwei Tian is with the School of Software, Xinjiang University, Urumqi 830046, China (e-mail: tianshengwei@163.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2023.3252807.

Digital Object Identifier 10.1109/TCSVT.2023.3252807

1051-8215 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications/rights/index.html for more information.
mapping from channel slices of image features to those of structure tokens. We instantiate the extraction operation in both content-agnostic and content-related manners, resulting in three different implementations, namely point-wise extraction (PWE), self-slice extraction (SSE) and cross-slice extraction (CSE). The corresponding three variants are denoted as StructToken-CSE, StructToken-SSE and StructToken-PWE, respectively. Specifically, PWE and SSE apply point-wise convolution and channel-wise self-attention to the concatenation of the image feature and structure tokens, respectively. CSE performs channel-wise cross-attention between structure tokens and image feature, where the former is used as query and the latter is treated as key and value. Since the point-wise convolution kernel weights are fixed after training, the extraction in PWE is independent of the input image, i.e., content-agnostic. While, the attention mechanisms in SSE and CSE determine the mapping weights according to the similarity between channel slices, which are content-related. In addition, SSE and PWE contain the mapping between all the channel slice pairs of image feature and structure tokens, while the more efficient CSE only involves the one-way mapping of channel slices from image feature to structure tokens. Interestingly, we find that PWE shows more advantages compared with the other two counterparts under fewer extraction operations (more details please refer to Figure 3). Furthermore, benefiting from stronger modeling capability, SSE begins to show its superiority under greater challenges.

We evaluate our approach on three challenging semantic segmentation benchmarks under different backbones. For example, equipped with ViT-L/16 [15] as backbone, our StructToken achieves 54.18% mIoU on ADE20K [55], 82.07% mIoU on Cityscapes [12] and 49.07% mIoU on COCO-Stuff-10K [3] respectively, which outperforms the state-of-the-art methods. Our main contributions include:

- We propose a new paradigm for semantic segmentation, termed structure-aware extraction paradigm, which follows a similar mechanism as humans and emphasizes the critical effect of structural information.
- We present a network, named StructToken under our structure-aware extraction paradigm, and explore the different implementations of the extraction process.
- Extensive experiments verify the effectiveness of our approach and show the prospect of human-like segmentation paradigm.

II. RELATED WORK

A. Static Per-Pixel Classification Paradigm

Since Fully Convolutional Networks (FCN) [34] were proposed, per-pixel classification has dominated semantic segmentation. It classifies each pixel to a specific category via a fixed classifier (such as a 1×1 convolution), which is unchanged after the training process. The methods under this paradigm mainly focused on how to learn better representation for each pixel via context modeling and fusion. The early PSPNet [53] used a pyramid pooling module to make multi-scale context fusion. The DeepLab family [5, 6] introduced the dilated convolution to expand the receptive fields. DANet [17], DSANet [22], CCNet [23] and OCRNet [49] used non-local modules to model more precise context information. Reference [42] proposed a stage-aware feature alignment module to align and fusion of features between adjacent levels. Reference [39] proposed the Gaussian dynamic convolution to adaptive fusion context information. Reference [24] enhance the ability to locate object boundaries by cascaded CRFs. Reference [25] extract the non-rigid geometry features by deformable convolution. Reference [2] strengthens that connection to the same object through the object-level semantic integration module for more efficient integration of context information. In addition, STLNet [56] starts to consider the structural information of the image itself, by introducing the texture enhance module and pyramid texture feature extraction Module to model the structural properties of textures in images. However, STLNet does it by modeling or statistically contextualizing the information itself, and it is still a per-pixel classification paradigm. Recent work [32], [36], [43], [46], [50], [54] began to use transformer architecture to capture long-range context information.

B. Dynamic Per-Pixel Classification Paradigm

Compared with the static one, this paradigm dynamically generates classifiers for each category based on the image content. Specifically, it establishes the connection between the image content and the classifier through attention and facilitates the classifier to be more suitable for the current sample image through the concatenation of multiple blocks. Segmenter [38] employed the transformer to jointly process the patches and class embeddings (tokens) during the decoding
phase and let the class tokens perform matrix multiplication with the feature map to produce the final score map. MaskFormer [10] unified instance segmentation and semantic segmentation architecture by performing matrix multiplication between class tokens and feature maps and using a binary matching mechanism. Mask2Former [9] and K-Net [52] used learned semantic tokens, which are equivalent to the class tokens, to replace 1 × 1 convolution kernels and used binary matching to unify semantic, instance, and panoptic segmentation tasks.

However, both of these two paradigms ignore the structural information of each category, which is critical in the human decision-making mechanism. Inspired by the segmentation discrimination mechanism of the human brain, we aim to explore a new paradigm to focus more on how to use the structural information as a cue for semantic segmentation task.

III. METHOD

In this section, we first present the overall framework of our StructToken. Then, we give the details about two basic components in each block of the decoder, the interaction module and the feed-forward network, respectively.

A. Framework

The overall framework of our StructToken is shown in Figure 2. Given an input image \( I \in \mathbb{R}^{3 \times h \times w} \), we first use a single-scale backbone (such as ViT [15]) to generate the feature map \( F \in \mathbb{R}^{C \times H \times W} \). Here \( C \) is the channel number, and \((h, w)\) and \((H, W)\) represent the spatial size (height and width) of the input image and feature map, respectively. Then, the feature map \( F \) and structure tokens \( S \in \mathbb{R}^{K \times H_s \times W_s} \) are sent to the decoder, where \( K \) means the total number of categories within the dataset. The structure tokens are learnable during training and fixed during inference, each of which contains the implicit structural information of a specific category. Note that, when \((H, W) \neq (H_s, W_s)\), the structure tokens are interpolated to the same spatial size of the feature map \( F \). The whole decoder contains \( L \) consecutive blocks. Each block consists of an extraction module and two feed-forward networks (FFN). The extraction module aims to extract the structural information from the feature map to structure tokens, and each of the resulting structure tokens can be regarded as a mask for each category. The two FFNs are used to refine the \( F \) and \( S \) via channel-wise projection respectively. For the last block, only the FFN for structure tokens is required because the feature map is not used in the following process.

Finally, a simple ConvBlock [21], including two 3 × 3 convolutions and a skip connection, is applied to the output structure tokens of the last block to further refine the segmentation results.

B. Extraction Module

As the structure tokens are learned from the whole dataset, the structural information in them is abstract and implicit, which entails further specification and refinement according to the current input image. Accordingly, the extraction module is designed to extract the structural information from the feature map to structure tokens. For comprehensiveness, we explore using content-agnostic convolution and content-related attention to implement the extraction operation, resulting in three variants: point-wise extraction (PWE), self-slice extraction (SSE), and cross-slice extraction (CSE), respectively. Specifically, PWE and SSE apply 1 × 1 convolution kernels; and channel-wise self-attention to the concatenation of the image feature and structure tokens, respectively. CSE performs channel-wise cross-attention between structure tokens and image feature, where the query comes from the former and the latter is treated as key and value. Since the extraction weights in PWE
are independent of the input image, it is content-agnostic, while the attention-based SSE and CSE are content-related. In addition, CSE can be regarded as a simplified version of SSE with only one-way mapping of channel slices from image feature to structure tokens. We provide the implementation details of these three variants as follows.

1) Cross-Slice Extraction: Considering that cross-attention is a well-known operation to aggregate information from one thing to another, which is highly compatible with the role of our interaction module. Thus, we utilize cross-attention to extract structural information from the feature map to structure tokens. Such process is named cross-slice extraction (CSE). The forward pass of CSE in the i-th block can be formulated as follows:

\[
Q_i = \Phi_q(S_i), \quad K_i = \Phi_k(F_i), \quad \mathcal{V}_i = \Phi_v(F_i),
\]

where, the projection layer \(\Phi_i\) is formulated as follows:

\[
S_{i+1} = \text{Softmax}(Q_i \times K_i^T \sqrt{\frac{1}{|C|}} \times \mathcal{V}_i),
\]

where, \(\mathcal{F}_i \in \mathbb{R}^{C \times H \times W}\) is the input feature map, and \(S_i \in \mathbb{R}^{K \times H \times W}\) denotes the structure tokens, which can be viewed as \(K\) tokens, each of which is a 2-dimension slice with height \(H\) and width \(W\). The query in cross-attention is generated by \(\Phi_q(S_i)\) here play a role of re-mapping each token in \(S_i\) and \(F_i\) via the same pattern. However, simply performing a fully-connected layer along the \(H \times W\) dimension on each token leads to the incompatibility for input images with arbitrary size as well as the multi-scale inference process. In order to solve this problem, we replaced the three fully-connected projections with local-connected ones. Specifically, the \(\Phi_i\) in Eq. (1) is formulated as follows:

\[
\Phi_{i} \in \{q, k, v\} \rightarrow \Phi_{i}(x) = \xi_{i}(\Phi_{i}(\xi_{i}(x))),
\]

where \(\Phi_{i}\) denotes a 3 \(\times\) 3 depth-wise convolution which maps each token locally, \(\xi_{i}\) and \(\xi_{i}\) are 1 \(\times\) 1 point-wise convolution to make each token have a preview of its counterparts.

2) Self-Slice Extraction: In this variant, we use self-attention to interact structure tokens and the feature map with each other. Specifically, in the forward process of the i-th block, it first concatenates the structure tokens \(S_i \in \mathbb{R}^{K \times H \times W}\) and the feature map \(F_i \in \mathbb{R}^{C \times H \times W}\) along the channel dimension,

\[
\mathcal{G}_i = \text{Concat}(S_i, F_i) \in \mathbb{R}^{(C+K) \times H \times W},
\]

Then, the self-attention is performed on \(\mathcal{G}_i\) to exchange information between structure tokens and feature map,

\[
Q_i = \Psi_q(\mathcal{G}_i), \quad K_i = \Psi_k(\mathcal{G}_i), \quad \mathcal{V}_i = \Psi_v(\mathcal{G}_i),
\]

where, the projection layer \(\Psi_{i} \in \{q, k, v\}\) share the same implementation of the \(\Phi_{i} \in \{q, k, v\}\) in CSE. \(Q_i, K_i\) and \(V_i\) have the same shape with \((C + K) \times H \times W\). Finally, the structure tokens \(S_{i+1} \in \mathbb{R}^{K \times H \times W}\) and feature map \(F_{i+1} \in \mathbb{R}^{C \times H \times W}\) are divided from the updated \(\mathcal{G}_i\) by directly split along the channel dimension,

\[
S_{i+1}, F_{i+1} = \text{Split}(\mathcal{G}_i).
\]

It can be found that the interaction in CSE is uni-directional \((S \rightarrow F)\) with only structure tokens being updated, while our SSE achieves a more comprehensive bi-directional interaction \((S \leftrightarrow F)\) in which both structure tokens and feature map are updated. Thus, SSE can be regarded as an extension of the CSE.

3) Point-Wise Extraction: As stated above, the attention map (with shape \(\mathbb{R}^{(C+K) \times (C+K)}\)) in SSE represents the aggregation weights for every slice of the concatenated feature, which is further used to filter out the unuseful information. Different from using dot-product to generate the aggregation weights, our point-wise extraction (PWE) is designed to directly learn the weights via a simple point-wise convolution layer. To be specific, in the forward process of the i-th decoder block, we also first concatenate the structure tokens \(S_i \in \mathbb{R}^{K \times H \times W}\) and the feature map \(F_i \in \mathbb{R}^{C \times H \times W}\) according to Eq. (4), resulting in \(\mathcal{G}_i \in \mathbb{R}^{(C+K) \times H \times W}\). Then, the interaction is performed via the point-wise convolution \(\Omega\), whose parameters are deemed the aggregation weights.

\[
\mathcal{G}_i = \mathbf{Y}(\mathcal{G}_i) \in \mathbb{R}^{(C+K) \times H \times W},
\]

\[
\mathcal{G}_i = \Omega(\mathcal{G}_i) \in \mathbb{R}^{(C+K) \times H \times W},
\]

where the projection layer \(\mathbf{Y}\) is implemented same as the \(\Psi\) and \(\Phi\) in Eq. (1) and Eq. (5). The \(\Omega\) denotes the point-wise convolution.

C. Feed-Forward Networks (FFN)

The traditional feed-forward networks (FFN) [40] is comprised of two consecutive fully connected layers to expand and shrink the channel dimension respectively. Considering that the FFN in our framework plays a role of refinement, we added a lightweight \(3 \times 3\) group convolution [29] between the original two fully connected layers to involve more local context (ablated in Table III).

IV. EXPERIMENTS

We first introduce the datasets and implementation details. Then, we compare our method with the recent state-of-the-arts on three challenging semantic segmentation benchmarks. Finally, comprehensive ablation studies and visual analysis are conducted to evaluate the effectiveness of our approach.

A. Datasets

ADE20K [55] is a challenging scene parsing dataset, which is split into 20210 and 2000 images for training and validation, respectively. It has 150 fine-grained object categories and diverse scenes with 1,038 image-level labels.

Cityscapes [12] carefully annotates 19 object categories of urban driveway landscape images. It contains 5K finely annotated images and is divided into 2975 and 500 images for training and validation, respectively. It is a high-quality dataset.
Comparison with the state-of-the-art methods on the ADE20K dataset. “SS” and “MS” indicate single-scale and multi-scale inference, respectively. 1) Means the ViT models trained from scratch on ImageNet-21k and fine-tuned on ImageNet-1k [37]. * Represents our implementation under the same settings as the official repo.

| Method        | Venue   | Backbone | GFLOPs | Params | mIoU (SS) | mIoU (MS) |
|---------------|--------|----------|--------|--------|-----------|-----------|
| FCN [34]      | CVPR15 | ResNet-101 | 276    | 69M    | 39.91     | 41.40     |
| EncNet [51]   | CVPR18 | ResNet-101 | 219    | 55M    | 44.65     |           |
| OCRNet [49]   | ECCV20 | HRNet-W48  | 165    | 71M    | 43.25     | 44.88     |
| CCNet [23]    | ICCV19 | ResNet-101 | 278    | 69M    | 43.71     | 45.04     |
| ANN [57]      | ICCV19 | ResNet-101 | 263    | 65M    | 45.24     |           |
| PSPNet [53]   | CVPR17 | ResNet-101 | 256    | 68M    | 44.39     | 45.35     |
| PPT [50]      | ECCV20 | ResNet-101 | -      | -      | -         | 43.90     |
| DeepLabV3+ [7]| ECCV18 | ResNet-101 | 255    | 63M    | 45.47     | 46.35     |
| STN-L [56]    | CVPR21 | ResNet-101 | -      | -      | -         | 46.48     |
| DMNet [19]    | ICCV19 | ResNet-101 | 274    | 72M    | 45.42     | 46.76     |
| ISNet [27]    | ICCV21 | ResNet-St-101 | - | - | - | 47.55 |
| DPT [36]      | ICCV22 | ViT-H/6d   | -      | -      | -         | 49.02     |
| DPT*          | ICCV21 | ViT-L/16†  | 324    | 338M   | 49.16     | 49.52     |
| UpperNet+     | ECCV18 | ViT-L/16†  | 710    | 354M   | 48.64     | 50.00     |
| SETR [54]     | CVPR21 | ViT-L/16†  | 214    | 310M   | 48.64     | 50.28     |
| MCBIC [26]    | ICCV21 | ViT-L/16†  | -      | -      | -         | 50.80     |
| SegFormer [46]| NeurIPS21 | MIT-B5 | 183    | 85M    | -         | 51.80     |
| SETR-MLA*     | CVPR21 | ViT-L/16†  | 214    | 310M   | 50.45     | 52.06     |
| UpperNet[33]  | ECCV18 | Swin-L     | 647    | 234M   | 52.10     | 53.50     |
| Segmenter [38]| ICCV21 | ViT-L/16†  | 380    | 342M   | 51.80     | 53.60     |
| StructToken-SSE| -     | ViT-L/16†  | 466    | 395M   | 52.82     | 54.00     |
| StructToken-CSE| -     | ViT-L/16†  | 398    | 350M   | 52.84     | 54.18     |
| StructToken-PWE| -     | ViT-L/16†  | 442    | 379M   | 52.95     | 54.03     |

COCO-Stuff-10K [3] is a significant scene parsing benchmark with 9000 training images and 1000 testing images. It has 171 categories.

B. Implementation Details

All the experiments are conducted on 8 NVIDIA Tesla V100 GPUs (32 GB memory per-card) with PyTorch implementation and mmsegmentation [35] codebase. We use ViT [15] as the backbone. During training, we follow the common setting using data augmentation such as random horizontal flipping, random resize, random cropping (512 × 512 for ADE20K and COCO-Stuff-10K, 768 × 768 for Cityscapes and 640 × 640 with ViT-L/16 for ADE20K), etc. As for optimization, we adopt a polynomial learning rate decay schedule; following prior works [33], we employ AdamW to optimize our model with 0.9 momenta and 0.01 weight decay; we set the initial learning rate as 2e-5. The batch size is set to 16 for all datasets. The total iterations are 160k, 80k, and 80k for ADE20K, Cityscapes, and COCO-Stuff-10K, respectively. For inference, we follow previous work [33], [54] to average the multi-scale (0.5, 0.75, 1.0, 1.25, 1.5, 1.75) predictions of our model. Interpolation operations are used for multi-scale inference. The slide-window test is applied here. The performance is measured by the widely-used mean intersection of union (mIoU) in all experiments. Considering the effectiveness and efficiency, we adopt the ViT-T/16 [15] as the backbone in the ablation study on ADE20K.

C. Comparisons With the State-of-the-Art Methods

1) Results on ADE20K: Table I reports the comparison with the state-of-the-art methods on the ADE20K validation set. From these results, it can be seen that our StructToken is +1.02%, +1.15% and +1.04% mIoU (52.82, 52.95 and 52.84 vs. 51.80) higher than Segmenter [38] with the same input size (640 × 640), respectively. When multi-scale testing is adopted, our StructToken is +0.4%, +0.43% and +0.58% mIoU (54.00, 54.03 and 54.18 vs. 53.60) higher than Segmenter, respectively. For ViT-T/16, as shown in Table V, our best results is +0.86% mIoU (42.99 vs. 42.13) higher than DPT [36] with the same input size (512 × 512). For ViT-S/16, our best result is +1.44% mIoU (48.89 vs. 47.45) higher than DPT. For ViT-B/16, our best result is +1.82% mIoU (51.82 vs. 50.00) higher than Segmenter. Furthermore, the larger the model is, the better StructToken performs.

2) Results on Cityscapes: Table II demonstrates the comparison results on the validation set of Cityscapes. The previous state-of-the-art method Segmenter with ViT-L/16 achieves 79.10% mIoU. Our StructToken is +0.54%, +0.91% and +0.95% mIoU (79.64, 80.01 and 80.05 vs. 79.10) higher than it, respectively. As to multi-scale inference, our method is +0.68%, +0.72% and +0.77% mIoU (81.98, 82.02, 82.07 vs. 81.30) higher than Segmenter, respectively.

3) Results on COCO-Stuff-10K: Table IV compares the segmentation results on the COCO-Stuff-10K testing set. It can be seen that our StructToken-SSE can achieve 49.07% mIoU, and our method is +4.18% mIoU higher than MCBIC [26] (49.07 vs. 44.89).
TABLE III
Ablation Study of Each Component in Our StructToken on ADE20K. “♦” Means the Basic Architecture of FFN, i.e., Two Consecutive Linear Layers, and “♦♠” Denotes the Above Basic FFN With a 3 × 3 Group Convolution Between Two Linear Layers to Enhance the Locality. All the Experiments Are Equipped With ViT-T/16 as the Backbone

| Interaction Module | FFN | ConvBlock | GFLOPs | Params | mIoU (SS) | mIoU (MS) |
|--------------------|-----|-----------|--------|--------|-----------|-----------|
| CSE                | ♦   |           | 6.74   | 8.5M   | 39.71     | 38.90     |
| CSE                | ♦♠  |           | 6.74   | 8.5M   | 38.17     | 38.85     |
| CSE                | ♦   | ✓         | 7.16   | 8.9M   | 38.01     | 39.29     |
| CSE                | ♦♠  | ✓         | 7.16   | 8.9M   | 39.12     | 40.23     |

TABLE IV
Comparison With the State-of-the-Art Methods on the COCO-Stuff-10K Dataset

| Method          | Venue | Backbone | mIoU (MS) |
|-----------------|-------|----------|-----------|
| PSPNet [53]     | CVPR1 | ResNet-101 | 38.86    |
| SVCNet [13]     | CVPR1 | ResNet-101 | 39.60    |
| DaNet [17]      | CVPR1 | ResNet-101 | 39.70    |
| EMANet [31]     | ICCV19| ResNet-101 | 39.90    |
| SpyGR [30]      | CVPR20| ResNet-101 | 39.90    |
| ACNet [14]      | ICCV19| ResNet-101 | 40.10    |
| OCRNet [49]     | ECCV20| HRNet-W48  | 40.50    |
| GINet [44]      | ECCV20| ResNet-101 | 40.60    |
| RecoNet [8]     | ECCV20| ResNet-101 | 41.50    |
| ISNet [27]      | ICCV21| ResNeSt-101| 42.08    |
| MCB1 [26]       | ICCV21| ViT-L/16   | 44.89    |
| StructToken-PWE| -     | ViT-L/16   | 48.24    |
| StructToken-CSE| -     | ViT-L/16   | 48.71    |
| StructToken-SSE| -     | ViT-L/16   | 49.07    |

Fig. 3. Comparison under different block numbers on ADE20K. Here, ViT-T/16 is used as the backbone.

TABLE V
Compare the Performance of ViT Variants on the ADE20K Dataset

| Method         | Backbone | GFLOPs | Params | mIoU (SS) | mIoU (MS) |
|----------------|----------|--------|--------|-----------|-----------|
| Segmenter      | ViT-T/16 | 6      | 7M     | 38.10     | 38.80     |
| UpperNet       |          | 35     | 11M    | 38.93     | 40.19     |
| SERT-MILA      |          | 10     | 11     | 39.88     | 41.09     |
| DPT            |          | 104    | 17M    | 40.82     | 42.13     |
| StructToken-CSE|         | 7      | 9M     | 39.12     | 40.23     |
| StructToken-SSE|         | 13     | 14M    | 40.81     | 42.24     |
| StructToken-PWE|         | 10     | 12M    | 41.87     | 42.99     |
| UpperNet       | SERT-MILA| 140    | 42M    | 45.53     | 46.14     |
| SERT-MILA      |          | 21     | 27M    | 44.85     | 46.30     |
| DPT            |          | 20     | 22     | 45.00     | 46.90     |
| StructToken-CSE|         | 118    | 36M    | 46.37     | 47.45     |
| StructToken-SSE|         | 23     | 30M    | 45.86     | 47.44     |
| StructToken-PWE|         | 37     | 41M    | 47.11     | 49.07     |
| UpperNet       | SERT-MILA| 292    | 128M   | 46.58     | 47.47     |
| SERT-MILA      |          | 171    | 110M   | 47.20     | 47.86     |
| DPT            |          | 65     | 92M    | 48.21     | 49.32     |
| StructToken-CSE|         | 81     | 107M   | 49.00     | 50.00     |
| StructToken-SSE|         | 86     | 113M   | 49.51     | 50.87     |
| StructToken-PWE|         | 123    | 142M   | 50.72     | 51.85     |
| UpperNet       | SERT-MILA| 105    | 132M   | 50.92     | 51.82     |

are added, which is +1.41% mIoU (39.12 vs. 37.71) higher than the base model, and +1.33 % mIoU (40.23 vs. 38.90) for multi-scale inference.

2) Number of Blocks: Figure 3 shows the comparison among StructToken-CSE, StructToken-SSE and StructToken-PWE under different block numbers. It can be seen that the performance of all the variants presents an upward trend with the increase of block number. For the trade-off between performance and computation complexity and the number of parameters, we choose to use 4 blocks by default for all the variants, which also means that the performance of our StructToken in Table I,II and IV are lower than its upper bound. Interestingly, SSE and CSE with more flexible content-related attention operation perform worse than the content-agnostic PWE, and performance gap between them narrows with the increase of block number. This may be attributed to the more inflexible convolution operation easier to learn, while the attention operation need more blocks to show its strengths.

3) Comparison of CSE, SSE, and PWE: We compare these three variants from the following three aspects: (a) The complexity of scenarios. As can be seen from Table I, II and IV, StructToken-PWE tends to perform better under small dataset and simple scenario (e.g., cityscapes with 19 categories). For the moderately complex scenarios (such as ADE20k with 150 classes), SSE, CSE and PWE have similar performance, in addition CSE saves 18% GFLOPs compared to SSE. However, as the scenario becomes more complicated (e.g., COCO-Stuff-10K with 171 categories), content-related attention (i.e., SSE and CSE) begins to show its strengths, having benefited from dynamic modeling. The SSE with greater complexity performs better in this case. Compared to SSE and CSE, while PWE performs poorly on larger datasets, it performs well on smaller ones. (b) Strength of backbone. As shown in Table V, when using ViT-T/16 [15] as the backbone, StructToken-PWE surpasses CSE and SSE counterparts by...
a large margin, with +2.75% and +1.06% mIoU respectively. As the backbone gets stronger, such performance gap gradually narrows (StructToken-PWE is only +0.2% mIoU higher than StructToken-SSE), and the content-relevant SSE gradually shows its advantages. In addition, content-relevant attention is more dependent on the features extracted by the backbone, and the richer the features, the better the performance. (c) Number of decoder blocks. The quantitative results in Figure 3 and qualitative visualization in Figure 4 show that PWE has a stronger advantage with fewer decoder blocks. The performance gap is reduced with the increase of block number.

E. Visual Analysis
To better understand the mechanism of our paradigm, we visualize the evolution of structure tokens with progressive extraction operations to show how it works. Figure 4 shows two examples sampled from ADE20K dataset, containing “car” and “door” respectively. We compare the three extraction methods in the first three rows of each example, where the 2nd column denotes the learned structure token slice corresponding to a specific category (“car” or “door”) and the 3rd ~6th columns represent its updated results in 1st ~4th blocks. From the 2nd column, we can find that the structure token learned from dataset is relatively abstract. It does not present an obvious object pattern, which is also understandable because of the diversity of objects in each category. The 3rd ~6th columns show that the structural information in the structure token is more and more obvious with the gradual extraction operations. In addition, StructToken-PWE presents a rough object outline after the first block (3rd column), while such phenomenon is much more ambiguous in the other two counterparts, which means that PWE can extract information from the image feature much faster. In contrast, the extraction process of content-related SSE seems the slowest.

We further visually compare our paradigms and per-pixel classification paradigms to better understand the differences in how they work. For better comparison, we instantiate a model following the per-pixel classification paradigm as a counterpart, which is more aligned with our paradigm. Specifically, we first apply a 1 × 1 convolution on the backbone output feature to project the channel number to the category number, then use four residual blocks [20] to transform the feature map, followed by a 1 × 1 convolution to generate
the final segmentation result. So the feature map output of each residual block has a similar meaning to structure tokens, i.e., each slice contains the structure information of a specific category. But their difference is that the structural information in the per-pixel classification paradigm only comes from the current input image, while the structural information in structure tokens is the prior knowledge learned from the dataset. In Figure 4, the 4th row of each example shows the visualization of the feature slice corresponding to the specific category from each residual block output. We can find that even though the per-pixel classification paradigm is similar with our paradigm in the final mIoU and output feature of the segmentation head, the feature map after each block presents a completely different pattern compared with the structure tokens. From 1st to 3rd rows, we can see the clear structure of the “car” and “door” categories in the structure tokens. In contrast, in the 4th row, we can only see the blurry structure or even no structure of the semantic class until the 1 x 1 convolution transform the feature maps to per-pixel classification score map. Such more explicit structure information provides strong evidence of the strength of our paradigm in retaining structural information.

V. CONCLUSION

In this paper, we propose a new paradigm different from the per-pixel classification, termed structure-aware extraction. The classical per-pixel classification methods only focus on learning better pixel representations or classification kernels while ignoring the structural information of objects, which is critical to human decision-making mechanism. In contrast, structure-aware extraction has a good ability to extract structural features. Specifically, it generates the segmentation results via the interactions between a set of learned structure tokens and the image feature, which aims to progressively extract the structural information of each category from the feature. We hope this work can bring some fundamental enlightenment to semantic segmentation and other tasks.

REFERENCES

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, “Augmented reality meets deep learning for car instance segmentation in urban scenes,” in Proc. Brit. Mach. Vis. Conf., vol. 1, 2017, p. 2.
[2] Q. Bo, W. Ma, Y.-K. Lai, and H. Zha, “All-higher-stages-in adaptive context aggregation for semantic edge detection,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 10, pp. 6778–6791, Oct. 2022.
[3] H. Caesar, J. Uijlings, and V. Ferrari, “COCO-stuff: Thing and stuff classes in context,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1209–1218.
[4] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, “GCNet: Non-local networks meet squeeze-excitation networks and beyond,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW), Oct. 2019, pp. 1–10.
[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 834–848, Apr. 2018.
[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 801–818.
[32] F. Lin, T. Wu, S. Wu, S. Tian, and G. Guo, “Feature selective transformer for semantic image segmentation,” 2022, arXiv:2203.14124.

[33] Z. Liu et al., “Swin transformer: Hierarchical vision transformer using shifted windows,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 1–11.

[34] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3431–3440.

[35] OpenMMLab Semantic Segmentation Toolbox and Benchmark, MM Segmentation Contributors, Shanghai, China, Jul. 2020.

[36] R. Ranftl, A. Bochkovskiy, and V. Kolm, “Vision transformers for dense prediction,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 1–10.

[37] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, “How to train your ViT? Data, augmentation, and regularization in vision transformers,” 2021, arXiv:2106.10270.

[38] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, “Segmenter: Transformer for semantic segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2021, pp. 7262–7272.

[39] X. Sun, C. Chen, X. Wang, J. Dong, H. Zhou, and S. Chen, “Gaussian dynamic convolution for efficient single-image segmentation,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 5, pp. 2937–2948, May 2022.

[40] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 1–11.

[41] V. Nekrasov, H. Chen, C. Shen, and I. Reid, “Fast neural architecture search of compact semantic segmentation models via auxiliary cells,” in Proc. CVPR, 2019, pp. 1–10.

[42] X. Weng, Y. Yan, S. Chen, J.-H. Xue, and H. Wang, “Stage-aware feature alignment network for real-time semantic segmentation of street scenes,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 7, pp. 4444–4459, Jul. 2022.

[43] S. Wu, T. Wu, F. Lin, S. Tian, and G. Guo, “Fully transformer networks for semantic image segmentation,” 2021, arXiv:2106.04108.

[44] T. Wu et al., “GINet: Graph interaction network for scene parsing,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 34–51.

[45] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, “Unified perceptual parsing for scene understanding,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 418–434.

[46] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “SegFormer: Simple and efficient design for semantic segmentation with transformers,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 12077–12090.

[47] Y. Li et al., “Learning dynamic routing for semantic segmentation,” in Proc. CVPR, 2020, pp. 1–10.

[48] M. Yin et al., “Disentangled non-local neural networks,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 191–207.

[49] Y. Yuan, X. Chen, and J. Wang, “Object-contextual representations for semantic segmentation,” in Proc. 16th Eur. Conf. Glasgow, U.K.: Springer, Aug. 2020, pp. 173–190.

[50] D. Zhang, H. Zhang, J. Tang, M. Wang, X. Hua, and Q. Sun, “Feature pyramid transformer,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2020, pp. 323–339.

[51] H. Zhang et al., “Context encoding for semantic segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 7151–7160.

[52] W. Zhang, J. Pang, K. Chen, and C. C. Loy, “K-Net: Towards unified image segmentation,” in Proc. NeuIPS, 2021, pp. 1–13.

[53] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2881–2890.

[54] S. Zheng et al., “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers,” in Proc. CVPR, 2021, pp. 1–10.

[55] B. Zhou et al., “Semantic understanding of scenes through the ADE20K dataset,” Int. J. Comput. Vis., vol. 127, no. 3, pp. 302–321, Mar. 2020.

[56] L. Zhu, D. Ji, S. Zhu, W. Gan, W. Wu, and J. Yan, “Learning statistical texture for semantic segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 12357–12364.

[57] Z. Zhu, M. Xu, S. Bai, T. Huang, and X. Bai, “Asymmetric non-local neural networks for semantic segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 593–602.