SeMask: Semantically Masked Transformers for Semantic Segmentation

Jitesh Jain[1*] Anukriti Singh[1] Nikita Orlov[2]
Zilong Huang[1] Jiachen Li[1] Steven Walton[1] Humphrey Shi[1,2]
[1]SHI Labs @ Georgia Tech & UIUC & Oregon [2]Picsart AI Research (PAIR)

Abstract

Finetuning a pretrained backbone in the encoder part of an image transformer network has been the traditional approach for the semantic segmentation task. However, such an approach leaves out the semantic context that an image provides during the encoding stage. This paper argues that incorporating semantic information of the image into pretrained hierarchical transformer-based backbones while finetuning improves the performance considerably. To achieve this, we propose SeMask, a simple and effective framework that incorporates semantic information into the encoder with the help of a semantic attention operation. In addition, we use a lightweight semantic decoder during training to provide supervision to the intermediate semantic prior maps at every stage. Our experiments demonstrate that incorporating semantic priors enhances the performance of the established hierarchical encoders with a slight increase in the number of FLOPs. We provide empirical proof by integrating SeMask into Swin Transformer and Mix Transformer backbones as our encoder paired with different decoders. Our framework achieves impressive performance of 58.25% *mIoU on the ADE20K dataset with SeMask Swin-L backbone and improvements of over* 3% *in the mIoU metric on the Cityscapes dataset. The code is publicly available on https://github.com/Picsart-AI-Research/SeMask-Segmentation.*

1. Introduction

Semantic Segmentation aims to perform dense prediction for labeling each pixel in an image corresponding to the class that the pixel represents. Transformer-based vision networks [16, 43] have outperformed Convolutional Neural Networks on the image-classification task [30]. In modern times, transformer backbones have shown impressive performance when transferred to downstream tasks like semantic segmentation [2, 23, 35].

Most of the architectural designs in vision transformers approach the problem in either of the two ways: (i) Use an existing pretrained backbone as an encoder and transfer it to downstream tasks using pre-existing standard decoders such as, Semantic FPN [29] or UperNet [48]; OR (ii) design a new encoder-decoder network where the encoder is pretrained on ImageNet for the semantic segmentation task. Both of these ways, as mentioned earlier, involve finetuning the encoder backbone on the segmentation task. Finetuning from a large-scale dataset help early attention layers to incorporate local information at lower layers of the transformers [38]. However, it can still not harness the semantic context during finetuning due to the relatively smaller size of the dataset and a change in the number and nature of semantic classes from classification to the segmentation task. Hierarchical vision transformers [35, 49] tackle the problem with progressive downsampling of features along the stages, although they still lack the semantic context of the image.

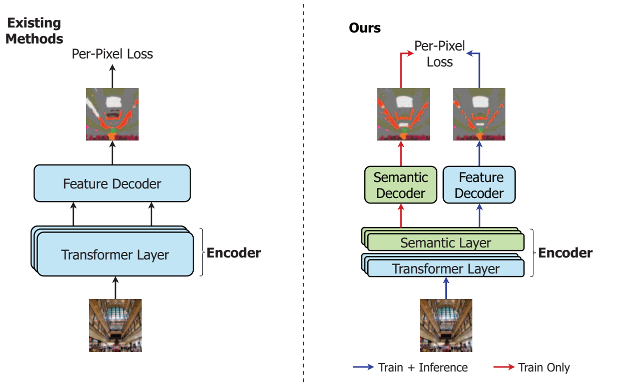


Figure 1: Comparison between popular transformer-based network for segmentation (left) and SeMask (right). In contrast to most existing methods ([35] in above figure) that directly use the pretrained backbones without any changes, SeMask uses semantic priors in the encoder backbones by adding an additional semantic layer; this simple change significantly improves performance.

Liu et al. [35] introduced the Swin Transformer, which constructs hierarchical feature maps making it compatible as a general-purpose backbone for major downstream vision tasks. [10] proposed to use two attention: globally subsampled and locally sub-samples on top of PVT [45] and CPVT [11] for effective segmentation. Xie et al. [49] further modified the hierarchical transformer encoder by mak-

[*] Work partially done during JJ's internship at PAIR.

ing it free from positional-encoding and thus robust to different resolutions as generally found in the segmentation task. All these works modified the encoders to make them work better for downstream tasks like segmentation and achieved success to an impressive extent. Still, they did not pay attention to capturing the semantic-level contextual information of the whole image. A lack of semantic contextual information leads to sub-optimal segmentation performance, especially in the case of small objects where those get merged with the boundaries of the larger categories, leading to wrong predictions. Recently, [41] tried to tackle this issue by designing a pure transformer-based decoder that jointly processes the patch and class embedding. However, it does not perform efficiently for tiny variants and fails with hierarchical architectures leading to sub-optimal performance when used with major transformer backbones like Swin [35], and Twins [10] transformers.

Jin et al. in [28] proposed ISNet to model the image level contextual information along with semantic level contextual information by introducing the SLCM and ILCM modules in the decoder structure. However there is still a caveat: ISNet is a CNN based method and only focuses on the decoder part of the network, leaving out the encoder unchanged.

To address the issues mentioned above, we propose the SeMask framework that incorporates semantic information into hierarchical vision transformer architectures and augments the global feature information captured by the transformers with the semantic context. The existing frameworks formulate the architecture as an encoder-decoder structure with transformers pretrained on ImageNet [30] acting as the encoders and using a specialized decoder for semantic segmentation. In contrast to directly using the hierarchical transformers as a backbone, we insert a Semantic Layer after the Transformer Layer at each stage in the backbone, giving us the SeMask version of the backbone as illustrated in Fig. 1. We use a lightweight semantic decoder to accumulate the semantic maps from all the stages, and a standard decoder like Semantic-FPN [29] for the main per-pixel prediction. The added semantic modeling with feature modeling throughout the encoder helps us improve the performance of the semantic segmentation task. In Sec. 4, we integrate the proposed SeMask block into the Swin Transformer [35] and Mix Transformer [49] backbones. Our experimental results show considerable improvement in semantic segmentation for both backbones on two different datasets. To summarize, our contributions are three fold:

- To the best of our knowledge, we are the first to study the effect of adding semantic context to pretrained transformer backbones for the semantic segmentation task. Furthermore, we introduce a SeMask Block which can be plugged into any existing hierarchical vision transformer. We provide empirical evidence by integrating SeMask into Swin-transformer [35] and Mix-Transformer [49], and achieving considerable performance improvement.
- We also propose to use a simple semantic decoder for aggregating the semantic priors from different stages of the encoder. The semantic priors receive supervision from the ground truth using a per-pixel cross-entropy loss.
- Lastly, we provide an in-depth analysis of the SeMask Block's effect on two different datasets: ADE20K and Cityscapes. We achieve impressive performance of 58.25% mIoU on the ADE20K dataset and an improvement above 3% on the Cityscapes dataset.

2. Related Work

2.1. Semantic Segmentation

Semantic segmentation broadly formulates to a dense per-pixel classification task. The seminal work of FCN [36] introduced the use of deep CNNs, removing fully connected layers to tackle the segmentation task. Several following works [1, 32, 39] were built upon the same idea of using the encoder-decoder architecture. [4] introduced the use of atrous convolutions inside the DCNN to tackle the signal downsampling issue. Later, various works focused on the aggregating long-range context in the final feature map: ASPP [5–7] uses atrous convolutions with different dilation rates; PPM [51] uses pooling with different kernel sizes.

The recent DCNN based models focus on efficiently aggregating the hierarchical features from a pretrained backbone based encoder with specially designed modules: [40, 42, 47] introduce attention modules in the decoder; [17, 24] use different forms of non-local blocks [46]; [31] proposes a novel FAM module to solve the misalignment issue using semantic flow; AlignSeg [25] proposes aligned feature aggregation module and aligned context modeling module to make contextual features be better aligned. [53] uses a segmentation shelf for better information flow. In this work, we also follow the established direction to use a pretrained backbone and aggregating the hierarchical features [35] using the Semantic-FPN [29] decoder.

2.2. Transformers for Segmentation

After being heavily used in Natural Language Processing field, transformer [44] based models have gained popularity for various computer vision tasks since the introduction of ViT [16] for image classification [16, 19, 26, 43]. SETR used ViT [16] as an encoder and two decoders based upon progressive upsampling and multi-level feature aggregation. SegFormer [49] proposed to use a hierarchical pyramid vision transformer network as an encoder with an MLP based decoder to obtain the segmentation mask. Segmenter [41] designed mask transformer as a decoder, which uses learnable class-map tokens to enhance decoding performance.

MaskFormer [9] defines the problem of per-pixel classification from a mask classification point of view, creating an all-in-one module for all segmentation tasks. Mask2Former [8] further evolves masked attention to solve panoptic, instance and semantic segmentation tasks in one framework. Most recent transformer-based segmentation frameworks [15, 35] are based on finetuning a pretrained hierarchical backbone as an encoder, and standard decoders like Semantic-FPN and UperNet [29, 48] to the segmentation task. In this work, we follow the same paradigm and, in addition, propose a framework to enhance the finetuning ability of the pretrained vision transformer backbone. Note that there is also recent concurrent work like SwinV2 [34] that reaches better performance on the ADE20k benchmark by using improved and giant backbones (e.g. SwinV2-G with 3.0 billion parameters, which is not released publicly). That is out of the scope of this work and we follow the current practice mainly based on Swin-L backbone. Theoretically, we can get even better performance if we apply our approach to such giant models.

2.3. Semantic Context in Segmentation

Zhang et al. proposed the Context Encoding Module in [50] which captures the global semantic context along with a feedback loop to balance the importance of classes in the features extracted by a ResNet backbone [20]. More recently, [27, 28] focus on capturing and integrating the semantic-level contextual information along with the image-level context with specially designed decoders which shows significant improvement in DCNN based methods. Each of these works captures the semantic context after the encoding stage based on the extracted features and not the encoder's ability to capture the semantic features.

In this work, we argue that semantic information is lost during the encoding stage and hence, propose a framework to capture semantic information which can be plugged into any pretrained vision transformer backbone network.

3. Method

An overview of our architecture with Swin-Transformer [35] backbone is shown in Fig. 2. The RGB input image, size $H \times W \times 3$, is first split into non-overlapping patches of size 4×4. The smaller size of the patch supports dense prediction in segmentation. These patches act as tokens and are given as input to the hierarchical vision transformer encoder, which is the Swin-Transformer [35] in our architecture. The encoding step consists of four different stages of hierarchical feature modeling. Every stage during the encoding step consists of two layers: The transformer layer, which is N_A number of Swin Transformer blocks (Fig. 3a) stacked together and Semantic Layer with N_S number of SeMask Attention blocks (Fig. 3b). We collectively refer to the Transformer Layer and Semantic Layer at each stage as our SeMask Block. The patch tokens pass through each stage at $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ of the original image resolution for the feature maps and intermediate semantic-prior maps extraction.

In the encoder part of the network, the Semantic Layer takes in features from the Transformer Layer as inputs and returns the intermediate semantic-prior maps and semantically masked features (Fig. 3b). When we plug the SeMask Attention Block into other hierarchical vision transformers, the Transformer Layer consists of attention blocks corresponding to the specific backbone, like Efficient-Self Attention-based Transformer Layer for the Mix Transformer [49] backbone. The semantically masked features from each stage are aggregated using the semantic-FPN [29] decoder for producing the final dense-pixel prediction. Moreover, the semantic-prior maps from all the stages are aggregated using a lightweight upsample & sum operation-based semantic decoder to predict the semantic-prior for the network during training. Both decoders' outputs are supervised using a weighted per-pixel cross-entropy loss. These additional semantic-prior maps greatly assist the feature extraction and eventually improve the performance on the semantic segmentation task.

3.1. SeMask Encoder

Each stage in our encoder consists of two layers: the Transformer Layer and the Semantic Layer. The transformer layer is composed of N_A Swin Transformer blocks stacked to extract image-level context information from the image. The semantic layer contains N_S SeMask Attention blocks stacked together to decouple semantic information from the features, producing semantic-priors and then updating the features with guidance from these semantic-prior maps.

Transformer layer. For the transformer layer, we adapt the hierarchical structure of Swin Transformer [35] which constructs hierarchical feature maps and has linear computational complexity to the image resolution. Before feeding the RGB image into the transformer layer in the first stage, we split it into non-overlapping patches of size is $4 \times 4 \times 3 = 48$. The first stage in the encoder has a linear embedding layer to change the feature dimension of the patch tokens. Inside each transformer layer, there are N_A shifted window attention blocks (Fig. 3a) that have linear computation complexity along with cross-window connections to handle non-overlapping regions, making the design effective for image-level feature modeling. For a hierarchical representation, we shrink our feature maps from $\frac{H}{4} \times \frac{W}{4}$ to $\frac{H}{8} \times \frac{W}{8}$ by patch merging layers for the next stage. This patch merging is iterated for the next stages to obtain a hierarchical feature map, with a resolution of $\frac{H}{2^{i+1}} + \frac{W}{2^{i+1}} \times C_i$ where $i \in \{1, 2, 3, 4\}$. X represents the input features inside the transformer layer block. And for computing self-attention in the transformer layer, X is transformed into: Q, K, V which are *query*, *key* and *value* matrices with same

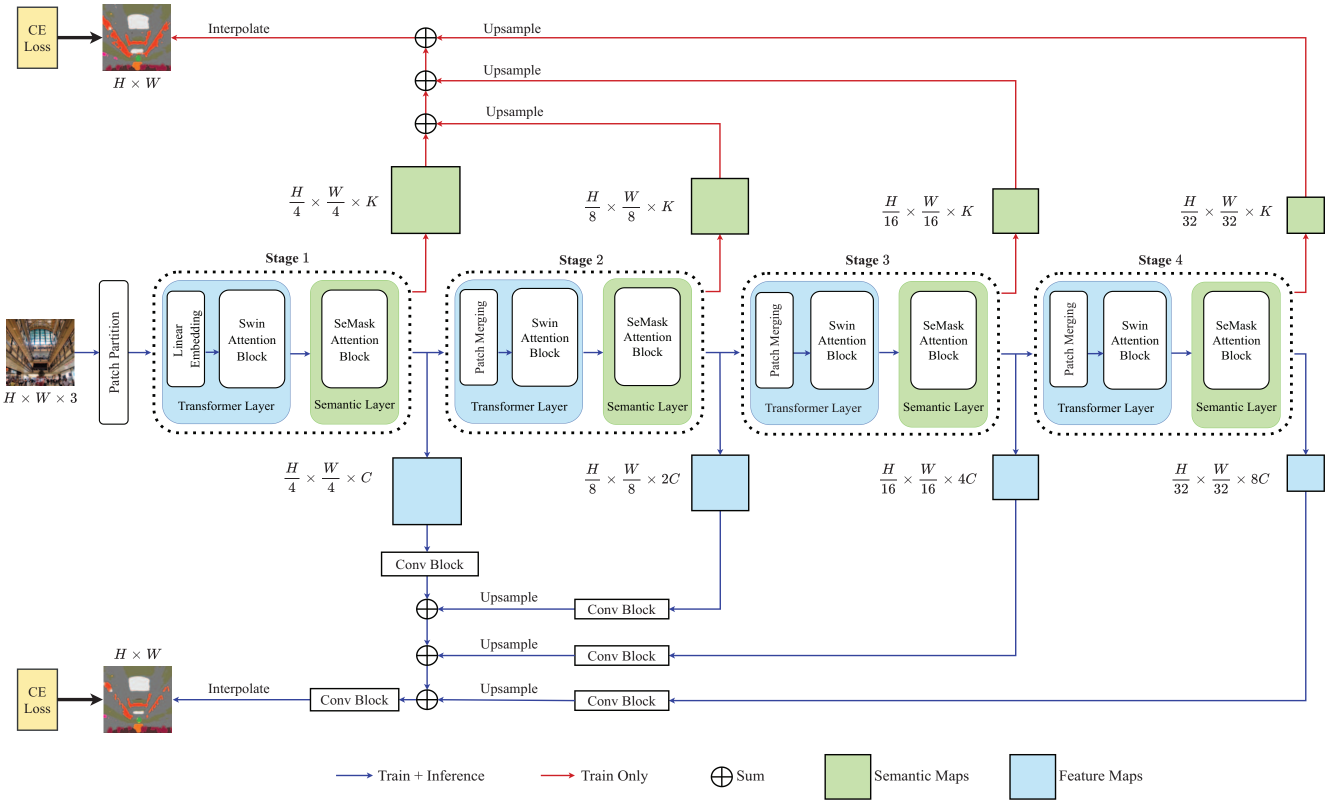


Figure 2: **SeMask Swin Semantic FPN Framework:** We add a Semantic Layer with N_S SeMask Blocks (Fig. 3b) after the Swin Transformer Layer to capture the semantic context in the encoder network. The Semantic Maps from the Semantic Layers at each stage are aggregated using a simple Upsample + Sum operation and passed through a weighted CE Loss to supervise the semantic context.

dimension of $N \times C$. Based on swin transformer, we also follow [3, 21, 22, 35, 37] to include a relative position embedding (RPE) where $RPE \in \mathbb{R}^{N \times N}$ and $N = M \times M$ is the length of the sequence with M = window size. The attention inside the Transformer Layer is calculated as:

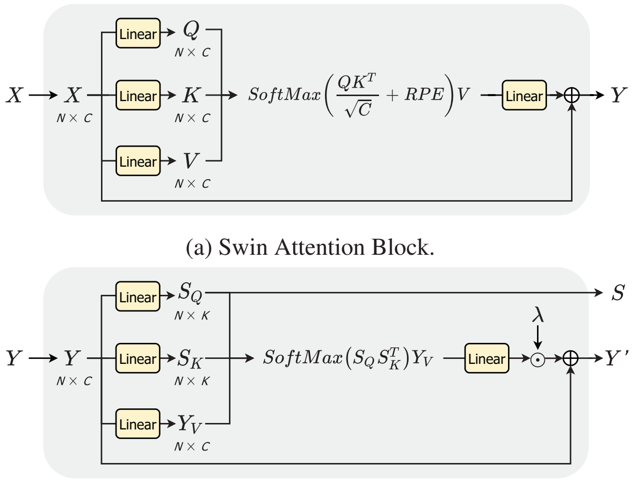
$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{C}} + RPE\right) V \quad (1)$$

The resulting feature Y from the Transformer Layer after the last Swin Transformer block then acts as an input to the subsequent semantic layer in the same stage as shown in Fig. 3.

Semantic Layer. The Semantic Layer follows the Transformer Layer at each stage of our hierarchical vision transformer. Unlike the Transformer Layer, the Semantic Layer's significance is in modeling the semantic context, which is used as a prior for calculating a segmentation score to update the feature maps based on guidance from the semantic nature present in the image. Inside each semantic layer, there are N_S SeMask attention blocks (Fig. 3b). Inspired by the shifted window-based division of the tokens for efficient computation cost, we also divide the input to our SeMask blocks into windows with cross-window connections before calculating the segmentation score using a single-head self-attention operation. The SeMask block is responsible for capturing the semantic context in our encoder. It updates the features from the transformer layer from the segmentation score providing guidance and giving a semantic-prior map for efficient supervision of the semantic modeling during training. SeMask attention block divides the features Y from the preceding transformer layer into three entities: Semantic Query (S_Q), Semantic Key (S_K), and Feature Value (Y_V). We get S_K and S_Q by projecting the features onto the semantic space. The dimension of both S_Q and S_K is $N \times K$ where K is equal to the number of classes, and the dimension of Y_V is $N \times C$ where C is the embedding dimension, $N = M \times M$ is the length of the sequence with M = window size which we set as equal to that used inside the transformer layer. S_Q returns the semantic map, and a segmentation score is calculated using S_K and S_Q. The score is passed through a softmax and is used to update Y_V as shown in Fig. 3b. This SeMask attention equation is expressed as follows:

$$\text{Score}(S_Q, S_K, Y_V) = \text{SoftMax}(S_Q S_K^T) Y_V \quad (2)$$

We perform a matrix multiplication between the feature



(a) Swin Attention Block.

(b) SeMask Attention Block.

Figure 3: **Attention Blocks.** N_A Shifted Window Self Attention Blocks, shown in Fig. 3a, are stacked inside each Transformer Layer and N_S SeMask Attention Blocks, shown in Fig. 3b, are stacked inside each Semantic Layer at every stage (Fig. 2). The output, Y, from the last Swin Attention Block, is fed to the first SeMask block in the Semantic Layer.

values and the segmentation score. The matrix product is later passed through a linear layer and multiplied with a learnable scalar constant λ, used for smooth finetuning. After a residual connection [20], we finally get the modified features, rich with semantic information which we call the **Se**mantically **Mask**ed features. The semantic queries S_Q are later used to predict the semantic-prior map.

3.2. Decoder

We use two decoders to aggregate the features and the semantic-prior maps respectively from the different stages in the encoder.

For aggregating the semantically masked features, we employ the popular Semantic-FPN decoder [29]. The Semantic-FPN fuses the features from different stages with a series of convolution, bilinear upsampling, and sum operations, making it efficient and straightforward as a segmentation decoder for our purpose. In addition, we use a lightweight semantic decoder during training to provide ground truth supervision to the semantic-prior maps at every stage of the encoder. As the semantic-prior maps have the channel dimension of K in each stage, we only employ a series of upsampling and sum operations to aggregate the maps with K being equal to the number of classes in the dataset. Lastly, the output from both the decoders is upscaled $\times 4$ to the resolution of the original image for the final predictions as shown in Fig. 2.

Backbone	**Window Size**	**Embedding Dim** (C)	**Blocks** (N_{T_B})	**Heads** (N_{T_H})	**#Params (M)**
Swin-T	7	[96, 192, 384, 768]	[2, 2, 6, 2]	[3, 6, 12, 24]	28
Swin-S	7	[96, 192, 384, 768]	[2, 2, 18, 2]	[3, 6, 12, 24]	50
Swin-B†	12	[128, 256, 512, 1024]	[2, 2, 18, 2]	[4, 8, 16, 32]	88
Swin-L†	12	[192, 384, 768, 1536]	[2, 2, 18, 2]	[6, 12, 24, 48]	197

Table 1: **Details of Swin Transformer variants.** The *Tiny* and *Small* variants are trained on ImageNet-1k and with 224×224 resolution. † stands for ImageNet-22k pre-training on 384×384 resolution images.

3.3. Loss function

To train our model's parameters, we calculate the total loss $\mathcal{L}_T$ as a summation of two per-pixel cross-entropy losses: $\mathcal{L}_1$ and $\mathcal{L}_2$. The loss $\mathcal{L}_1$ is calculated on the main prediction from the Semantic-FPN decoder and loss $\mathcal{L}_2$ is calculated on the semantic-prior prediction from our lightweight decoder. $\mathcal{F}$ contains the main prediction of the network and $\mathcal{S}$ denotes the semantic-prior prediction. We define our losses on $\mathcal{F}$ and $\mathcal{S}$ as follows:

$$\mathcal{L}_1 = \frac{1}{H \times W} \sum_{i,j} \mathcal{L}_{ce} \left(\mathcal{F}_{[*,i,j]}, \, \S \left(\mathcal{GT}_{[ij]} \right) \right). \tag{3}$$

$$\mathcal{L}_2 = \frac{1}{H \times W} \sum_{i,j} \mathcal{L}_{ce} \left(\mathcal{S}_{[*,i,j]}, \, \S \left(\mathcal{GT}_{[ij]} \right) \right). \tag{4}$$

$$\mathcal{L}_{\mathcal{T}} = \mathcal{L}_1 + \alpha \mathcal{L}_2 \tag{5}$$

Here, $\S$ denotes for converting the ground truth class label stored in $\mathcal{GT}$ into one-hot format, $\sum_{i,j}$ denotes that the summation is carried out over all the pixels of the $\mathcal{GT}$, and $\mathcal{L}_{ce}$ is the cross-entropy loss. We empirically set $\alpha = 0.4$ (check appendix for more details).

4. Experiments

We compare our approach with Swin Transformer [35], and Mix-Transformer [49] with extensive experiments to demonstrate the effectiveness of the SeMask framework. We also ablate the SeMask structure and confirm that providing a semantic-prior to mask out the features improves semantic segmentation performance. The experiments are performed on two widely used datasets: ADE20K [14] and Cityscapes [13]. We include more experimental results in the appendix proving that our method is dataset agnostic.

4.1. Datasets and metrics

ADE20K. [14] ADE20K is a scene parsing dataset covering 150 fine-grained semantic concepts and it is one of the most challenging semantic segmentation datasets. The training set contains 20,210 images with 150 semantic classes. The validation and test set contain 2,000 and 3,352 images respectively.

Cityscapes. [13] Cityscapes is an urban street driving dataset for semantic segmentation consisting of 5,000 images from 50 cities with 19 semantic classes. There are 2,975 images in the training set, 500 images in the validation set and 1,525 images in the test set.

Metrics. We report mean Intersection-over-Union ($mIoU$) over all classes.

Method	**Backbone**	**ADE20K**					**Cityscapes**				
(encoder + decoder)	(pretrained)	crop size	#param. (M)	FLOPs (G)	s.s. mIoU (%)	m.s. mIoU (%)	crop size	#param. (M)	FLOPs (G)	s.s. mIoU (%)	m.s. mIoU (%)
Swin-T FPN	Swin-T	512×512	33	38	41.48	42.89	768×768	33	81	71.81	73.74
SeMask-T FPN	SeMask Swin-T	512×512	35	40	**42.06** (+0.58)	**43.36** (+0.47)	768×768	34	84	**74.92** (+3.11)	**76.56** (+2.82)
Swin-S FPN	Swin-S	512×512	54	61	45.20	46.96	768×768	54	130	75.19	77.68
SeMask-S FPN	SeMask Swin-S	512×512	56	63	**45.92** (+0.72)	**47.63** (+0.67)	768×768	56	134	**77.13** (+1.94)	**79.14** (+1.46)
Swin-B FPN	Swin-B†	512×512	93	103	48.80	50.28	768×768	93	211	76.54	79.05
SeMask-B FPN	SeMask Swin-B†	512×512	96	107	**49.35** (+0.55)	**50.98** (+0.70)	768×768	96	217	**77.70** (+1.16)	**79.73** (+0.68)
Swin-L FPN	Swin-L†	640×640	204	343	50.85	52.95	768×768	204	444	78.03	79.53
SeMask-L FPN	SeMask Swin-L†	640×640	212	356	**51.89** (+1.04)	**53.52** (+0.57)	768×768	211	455	**78.53** (+0.50)	**80.39** (+0.86)

Table 2: **Ablation on Swin-Transformer varaints.** We provide a comparison of using SeMask Swin with Semantic-FPN [29] decoder on all 4 varaints on the **ADE20K-Val** and **Cityscapes-Val** dataset. We evaluate the models using both, the *single scale (s.s)* and *multi-scale (m.s.)* mIoU (↑). All models are trained for $80k$ iterations. The FLOPs are calculated for the given crop sizes using the script provided by the MMSegmentation [12] library.

Method	**Backbone**	**SA Block**	**SeMask Block**	**mIoU (%)**	**#Param (M)**
Swin-T FPN	Swin-T			41.48	33
Trans Swin-T FPN	Trans Swin-T	✓		41.42	36
SeMask-T FPN	SeMask Swin-T		✓	**42.06**	35

Table 3: **Ablation on Semantic Attention.** We prove the effectiveness of the *SeMask Block* by replacing it with a simple Single-Head Self Attention block which harms the performance on the *Tiny* variant.

Method	**Backbone**	λ	**mIoU (%)**	**#Param (M)**
SeMask-T FPN	SeMask Swin-T	✓	**42.06**	35
SeMask-T FPN	SeMask Swin-T	×	41.11	35
SeMask-S FPN	SeMask Swin-S	✓	**45.92**	56
SeMask-S FPN	SeMask Swin-S	×	45.00	56

Table 4: **Ablation on λ.** We support the critical claim of the learnable scalar constant: λ inside the SeMask Block by removing and recording the mIoU (↑).

Method	**Backbone**	**Auxiliary Loss**	**mIoU (%)**
Swin-T FPN	Swin-T		41.48
Swin-T FPN	Swin-T	✓	41.52
Query Swin-T FPN	Swin-T	✓	40.81
SeMask-T FPN	SeMask Swin-T		41.72
SeMask-T FPN	SeMask Swin-T	✓	**42.06**

Table 5: **Ablation on Auxiliary Loss.** We study the effect of the auxiliary loss on performance. Query Swin-T FPN uses the queries from the transformer layer for loss calculation. We observe that our SeMask performs the best.

4.2. Implementation details

Transformer models. For the encoder, we build upon the Swin Transformer [35] and consider the *Tiny*, Small, *Base* and *Large* variants as described in Tab. 1. The variation in *number of parameters* among the baselines is due to the *number of transformer blocks* (N_{T_B}) (Fig. 3a) and the *embedding dimension* (C) for each stage of the model. The *number of heads* (N_{T_H}) of a shifted window based multi-headed self-attention (SW-MSA) or Swin Transformer block varies from stage to stage. The hidden size of the MLP following SW-MSA is four times the embedding dimension at the corresponding stage. We also experiment with the MiT-B4 backbone variant of the Mix-Transformer [49] on the ADE20K [14] dataset.

In the following sections, we use an abbreviation to describe the model variant. For example, Swin-T denotes the *Tiny* variant. The backbones pretrained on ImageNet-22k [30] and with 384×384 resolution are denoted with a †: Swin-B†. All the other models are pretrained on ImageNet-1k and with 224×224 resolution.

Network Initialization. Our SeMask models are initialized with publicly available models. The *Tiny* and *Small* variants are pre-trained on ImageNet-1k with an image resolution of 224×224. The *Base* and *Large* variants are pretrained on ImageNet-22k with a resolution of 384×384. We keep the window size (M) fixed as in the pretrained models and finetune the models for the semantic segmentation task at higher resolution depending on the dataset. Following [35], we include relative position bias while calculating the attention scores. The decoders, described in Sec. 3.2 are initialized with random weights from a normal distribution [18].

Data augmentation. During training, we perform mean subtraction, scaling the image to a ratio randomly sampled from $(0.5, 0.75, 1.0, 1.25, 1.5, 1.75)$, random left-right flipping, and color jittering. We randomly crop large images and pad small images to a fixed size of 512×512 for ADE20K and 768×768 for Cityscapes. On ADE20K, we train our largest model Semask-L† FPN with a 640×640 resolution, matching the resolution used by the Swin-Transformer [35].

4.3. Ablation Studies

In this section, we ablate different variants of our SeMask framework. We investigate the model size, semantic attention, effect of the learnable scalar constant (λ) inside the *SeMask* block and the auxiliary loss. Unless stated otherwise, we use the Semantic-FPN [29] as our decoder for the main prediction and report results using single-scale (s.s.) inference on the ADE20K [14] val dataset.

Transformer size. We study the impact of transformers size on performance in Tab. 2 by experimenting with the four different Swin variants: *Tiny*, *Small*, *Base* and *Large* with $N_S = [1, 1, 1, 1]$ for all the experiments. Our method

Method	Backbone	Crop Size	mIoU (%)	MS mIoU (%)
CNN Backbones				
FCN [36]	ResNet-101	512 × 512	39.91	41.40
PSPNet [51]	ResNet-101	512 × 512	44.39	45.35
DLab.v3+ [7]	ResNet-101	512 × 512	45.47	46.35
Transformer Backbones				
SegFormer [49]‡	MiT-B4	512 × 512	48.46	49.76
Swin-L FPN [35]	Swin-L†	640 × 640	50.85	52.95
Seg-L-Mask/16 [41]	ViT-L/16†	640 × 640	51.80	53.56
Swin-L UPerNet [35]	Swin-L†	640 × 640	—	53.50
SwinV2-L UPerNet [34]*	SwinV2-L†	640 × 640	—	55.90
Swin-L MaskFormer [9]	Swin-L†	640 × 640	54.10	55.60
Swin-L Mask2Former [8]	Swin-L†	640 × 640	56.10	57.30
Swin-L MSFaPN-Mask2Former [8]$^\Vert$	Swin-L†	640 × 640	55.99	57.69
Swin-L FaPN-Mask2Former [8]	Swin-L†	640 × 640	56.40	57.70
SeMask SegFormer *(Ours)*	SeMask MiT-B4	512 × 512	50.01	51.07
SeMask-L FPN *(Ours)*	SeMask Swin-L†	640 × 640	51.89	53.52
SeMask-L MaskFormer *(Ours)*	SeMask Swin-L†	640 × 640	54.75	56.15
SeMask-L Mask2Former *(Ours)*	SeMask Swin-L†	640 × 640	56.41	57.52
SeMask-L FaPN-Mask2Former *(Ours)*	SeMask Swin-L†	640 × 640	56.88	**58.25**
SeMask-L MSFaPN-Mask2Former$^\Vert$ *(Ours)*	SeMask Swin-L†	640 × 640	**57.00**	**58.25**

Table 6: **Comparison on ADE20K-Val.** We report both single-scale (s.s.) and multi-scale (m.s.) mIOU (↑) on *ADE20K Val set*. ‡ We use the results from the MMSegmentation [12] library due to the reproducibility issues with the official SegFormer repo [link]. $^\Vert$We develop an MSFaPN network based on the changes done in FaPN [23] to the BasePixelDecoder [9]. *Note that we follow the convention and compare methods based on the Swin-L backbone and we currently do not consider giant models like SwinV2-G that have billions of parameters.

Method	Backbone	mIoU (%)	MS mIoU (%)
CNN Backbones			
PSANet [52]	ResNet-101	77.94	79.05
DeepLabV3+ [7]	Xception-71	-	79.55
CCNet [24]	ResNet-101	80.50	81.30
HRNetV2-OCR+PSA [33]	HRNetV2-W48	-	**86.95**
Transformer Backbones			
Seg-L-Mask/16 [41]	ViT-L/16†	79.10	81.30
Swin-L FPN [35]	Swin-L†	78.03	79.53
MaskFormer [9]	ResNet-101	78.50	80.30
Mask2Former [8]	Swin-L†	83.30	84.30
SeMask-L FPN *(Ours)*	SeMask Swin-L†	78.53	80.39
SeMask-L Mask2Former *(Ours)*	SeMask Swin-L†	**83.97**	**84.98**

Table 7: **Comparison on Cityscapes-Validation.** We report both single-scale (s.s.) and multi-scale (m.s.) mIOU (↑) on *Cityscapes Validation set*.

consistently improves over all the baseline variants with the improvement on the Cityscapes dataset being more impressive due to the fewer classes in the segmentation dataset creating a stronger prior.

We evaluate and record the mIoU scores for the baseline Swin models by training our networks using their publicly released code based on the MMSegmentation Library [12].

Semantic Attention. We study the impact of the semantic attention operation calculated inside the *SeMask Block* on performance in Tab. 3 by replacing the SeMask Block with a simple single-head self-attention block on the Swin-Tiny variant. It is evident that simple attention does not help improve the results proving the validity and effectiveness of our SeMask Block.

Learnable Constant (λ). We study the impact of λ on performance in Tab. 4, by removing it for the *Tiny* and *Small* variants. We observe that the inclusion of λ is potent to the success of the SeMask block as it acts as a tuning factor for the modified features, keeping the noise from weights' initialization in check. We also observe that $\lambda \in [0.05, 0.3]$ during inference for different stages in the encoder.

Ablation on Auxiliary Loss. We study the impact of the auxiliary CE Loss ($\mathcal{L}_2$) in Tab. 5. First, we add the extra supervision to the output of the transformer layer, which has a negligible effect on the performance. Since we use S_Q for calculating $\mathcal{L}_2$ in SeMask, we experiment with another baseline Query Swin-T FPN where we provide extra supervision to the queries inside the transformer, which shows a significant drop in performance. Thus, our SeMask-T FPN performs the best.

4.4. Main Results

ADE20K. We compare with several recently published methods. Using SeMask Swin-L† as the encoder and Mask2Former-MSFaPN as our decoder for the main prediction, we achieve scores of 57.00% and 58.25% on the single-scale and multi-scale mIoU metric, respectively. Following [35], our models were trained on 640×640 images. We also achieve competitive results with our SeMask Swin-L† backbone with Semantic-FPN to the Swin-

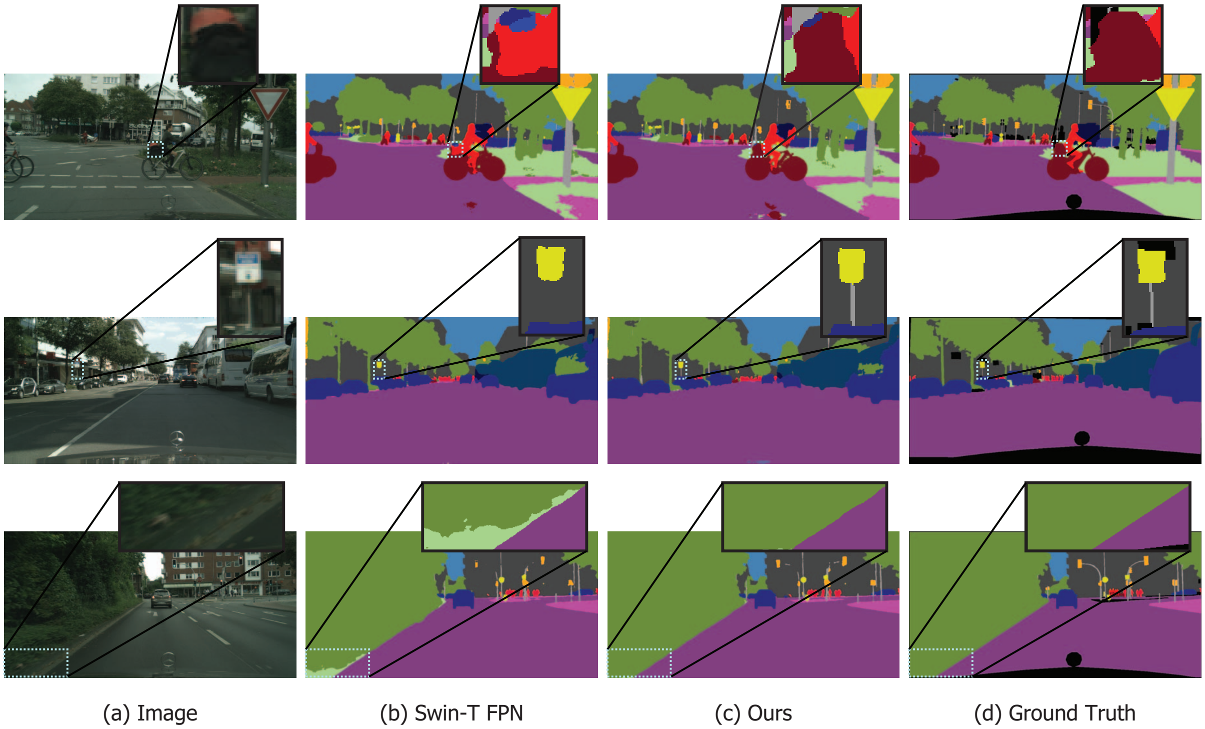


Figure 4: **Qualitative results on the Cityscapes validation set.** The dot-bordered boxes at the top show zoomed-in regions from the images for a more detailed look at the improvement using our **SeMask-T FPN**. Unlike Swin-T FPN, SeMask-T FPN does not wrongly segment the bicycle as part of the rider in the first row. For the second row, SeMask-T FPN can segment the pole. Similarly, in the third row, SeMask-T FPN segments the boundary in a better way than the baseline Swin-T FPN.

L† based UPerNet model as shown in Tab. 6.

We also integrate our SeMask into the MiT-B4 based SegFormer model [49] as shown in Tab. 6 and achieve an improvement of 1.55% on the single scale mIoU and 1.31% improvememt on the multi-scale mIoU metric scores. This supports our claim that SeMask can be plugged into any existing hierarchical vision transformer and show performance improvement.

Cityscapes. Tab. 7 reports the performance of SeMask on Cityscapes. Semask Swin-L† is competitive to other methods with SeMask Swin-L† Mask2Former achieving 84.98% mIoU. We train our SeMask-L Mask2Former on 512×1024 images following Mask2Former [8]. Furthermore, we achieve an impressive improvement of 3.11% s.s mIoU and 2.82% m.s mIoU with our SeMask-T FPN over its Swin-T FPN counterpart.

Qualitative results. Fig. 4 shows a qualitative comparison of Swin-T FPN and SeMask-T FPN on the Cityscapes dataset generated using the MMSegmentation library [12]. It is evident that SeMask-T FPN is able to generate better class-wise predictions than the Swin-T FPN. As shown in the second row in Fig. 4, we are able to segment the pole with our SeMask-T FPN, while Swin-T FPN fails to do so. Similarly in the third row, we are better able to segment the boundary.

5. Conclusion

This paper argues that directly finetuning off-the-shelf pretrained transformer backbones as encoders for semantic segmentation does not consider the semantic context tied to the images. We claim that adding a semantic prior to guide the encoder's feature modeling enhances the finetuning process for semantic segmentation. To support our claim, we propose the SeMask Block, which can be plugged into any existing hierarchical vision transformer and uses a semantic attention operation to capture the semantic context. We train and evaluate the proposed framework building on the Swin-Transformer [35] and Mix-Transformer [49] backbones-based networks and show a considerable improvement in the semantic segmentation performance on the Cityscapes and ADE20K dataset, with improvements above 3% on the Cityscapes dataset. We provide a comprehensive experimental analysis by applying SeMask to different backbone variants and achieving considerable performance improvement in every setting. As a direction for future research, it will be interesting to observe the effect of adding similar priors for other vision downstream tasks.

Acknowledgments. This material is based on work partially supported by the National AI Institute for Exceptional Education (Award #2229873) by the NSF and the Institute of Education Sciences, U.S. Department of Education.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv*, 2015. 2

[2] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv*, 2021. 1

[3] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*, 2020. 4

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 2

[5] 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. In *TPAMI*, 2017. 2

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

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2, 7

[8] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv*, 2021. 3, 7, 8

[9] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021. 3, 7

[10] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 2021. 1, 2

[11] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv*, 2021. 1

[12] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 6, 7, 8

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 5

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. Semantic understanding of scenes through the ade20k dataset. In *CVPR*, 2017. 5, 6

[15] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Nenghai Yu Weiming Zhang, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arxiv:preprint*, 2021. 3

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2

[17] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019. 2

[18] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *NeurIPS*, 2018. 6

[19] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv*, 2021. 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 5

[21] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018. 4

[22] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019. 4

[23] Shihua Huang, Zhichao Lu, Ran Cheng, and Cheng He. FaPN: Feature-aligned pyramid network for dense image prediction. In *ICCV*, 2021. 1, 7

[24] Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang. Ccnet: Criss-cross attention for semantic segmentation. In *TPAMI*, 2020. 2, 7

[25] Zilong Huang, Yunchao Wei, Xinggang Wang, Wenyu Liu, Thomas S Huang, and Humphrey Shi. Alignseg: Feature-aligned segmentation networks. In *TPAMI*, 2021. 2

[26] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. 2

[27] Zhenchao Jin, Tao Gong, Dongdong Yu, Qi Chu, Jian Wang, Changhu Wang, and Jie Shao. Mining contextual information beyond image for semantic segmentation. In *ICCV*, 2021. 3

[28] Zhenchao Jin, Bin Liu, Qi Chu, and Nenghai Yu. Isnet: Integrate image-level and semantic-level context for semantic segmentation. In *ICCV*, 2021. 2, 3

[29] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 1, 2, 3, 5, 6

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1, 2, 6

[31] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *ECCV*, 2020. 2

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

[33] Huajun Liu, Fuqiang Liu, Xinyi Fan, and Dong Huang. Polarized self-attention: Towards high-quality pixel-wise regression. *arXiv*, 2021. 7

[34] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv*, 2021. 3, 7

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 7

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. 4

[38] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *arXiv*, 2021. 1

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[40] Qi Song, Kangfu Mei, and Rui Huang. Attanet: Attention-augmented network for fast and accurate scene parsing. In *AAAI*, 2021. 2

[41] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 2, 7

[42] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv*, 2020. 2

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv*, 2020. 1, 2

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

[45] 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. In *ICCV*, 2021. 1

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

[47] Yu Wang, Quan Zhou, Jia Liu, Jian Xiong, Guangwei Gao, Xiaofu Wu, and Longin Jan Latecki. Lednet: A lightweight encoder-decoder network for real-time semantic segmentation. In *ICIP*, 2019. 2

[48] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 1, 3

[49] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. 1, 2, 3, 5, 6, 7, 8

[50] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. *arXiv*, 2018. 3

[51] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 7

[52] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. PSANet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018. 7

[53] Juntang Zhuang, Junlin Yang, Lin Gu, and Nicha Dvornek. Shelfnet for fast semantic segmentation. In *ICCV*, 2019. 2