Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers

Sixiao Zheng[1] Jiachen Lu[1] Hengshuang Zhao[2] Xiatian Zhu[3] Zekun Luo[4] Yabiao Wang[4]
Yanwei Fu[1] Jianfeng Feng[1] Tao Xiang[3,5] Philip H.S. Torr[2] Li Zhang[1]
[1]Fudan University [2]University of Oxford [3]University of Surrey [4]Tencent Youtu Lab [5]Facebook AI

CVPR VIRTUAL JUNE 19-25

SETR Project page: https://fudan-zvg.github.io/SETR/

Motivation and Contribution

Motivation:

- Most recent semantic segmentation methods adopt a FCN with an encoder-decoder architecture.
- Learning long-range dependency information is critical for semantic segmentation
- Latest efforts focuse on increasing the receptive field, atrous convolutions, inserting attention modules
- But all remain the FCN encoder-decoder architecture unchanged

Contribution:

- Reformulate the image semantic segmentation problem from a sequence-to-sequence learning perspective
- Offering an alternative to the encoder-decoder FCN model design.
- Provide a powerful segmentation model SETR
- Introduce three different decoder designs.
- Achieves new SOTA on ADE20K (50.28% mIoU), Pascal Context (55.83% mIoU) and competitive results on Cityscapes. Achieve the *first* position in the ADE20K test server leaderboard.

SEgmentation TRansformer (SETR)

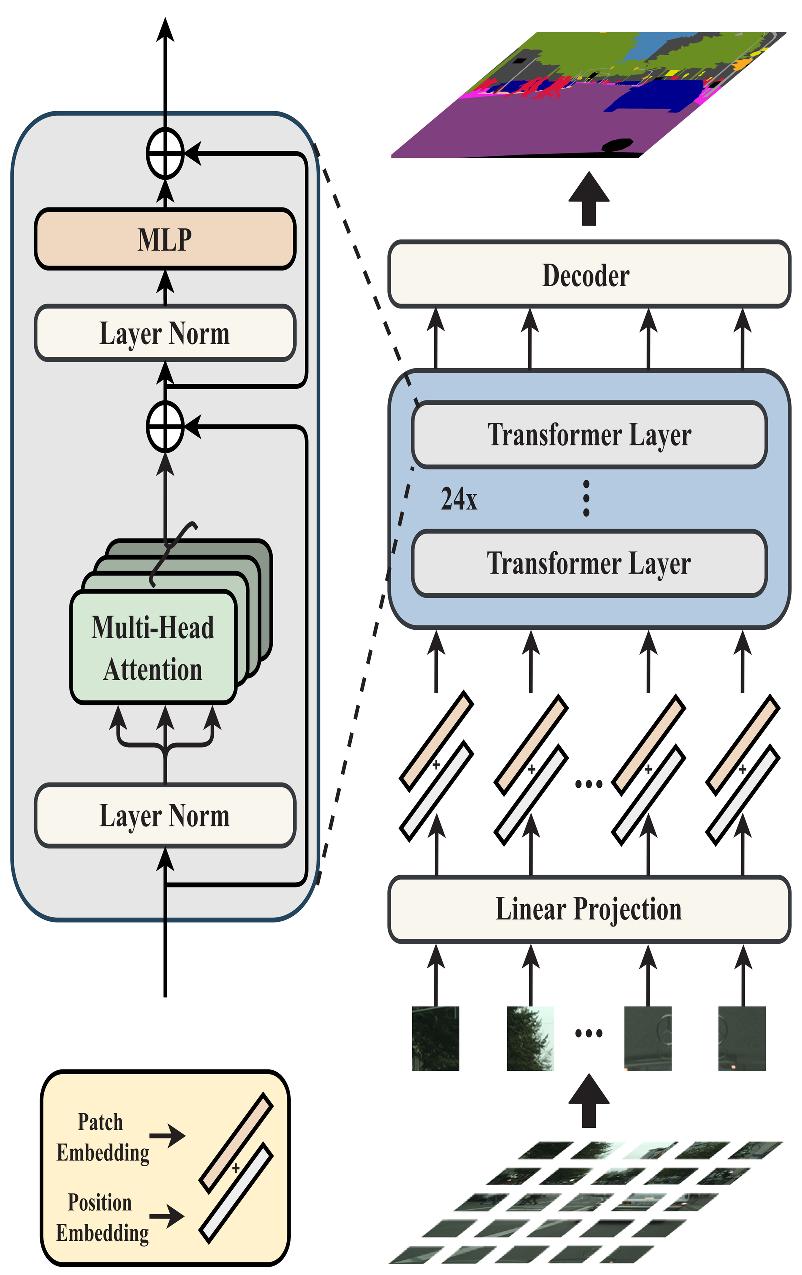

Figure 1. SETR

Image to sequence:

- Divide an image into a grid of patches uniformly, and then flatten it into a sequence.
- The vectorized patches are mapped into a 1D sequence of patch embeddings using a linear projection function.
- Add learnable position embeddings to the patch embeddings as the final input of the transformer encoder.

Transformer:

- A pure transformer based encoder is employed to learn feature representations.
- Each transformer layer has a global receptive field, solving the limited receptive field problem of existing FCN encoder once and for all.
- The transformer encoder consists of multi layers of multi-head self-attention (MSA) and Multilayer Perceptron (MLP) blocks.

Decoder designs

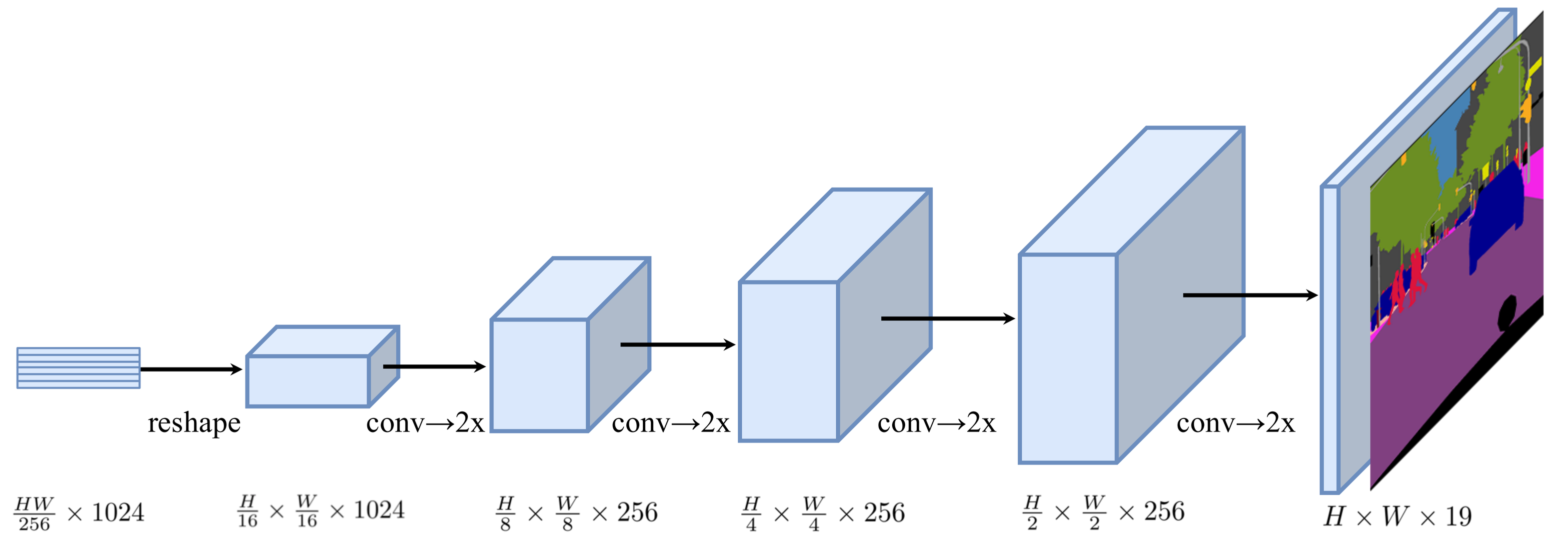

Figure 2. Progressive UPsampling

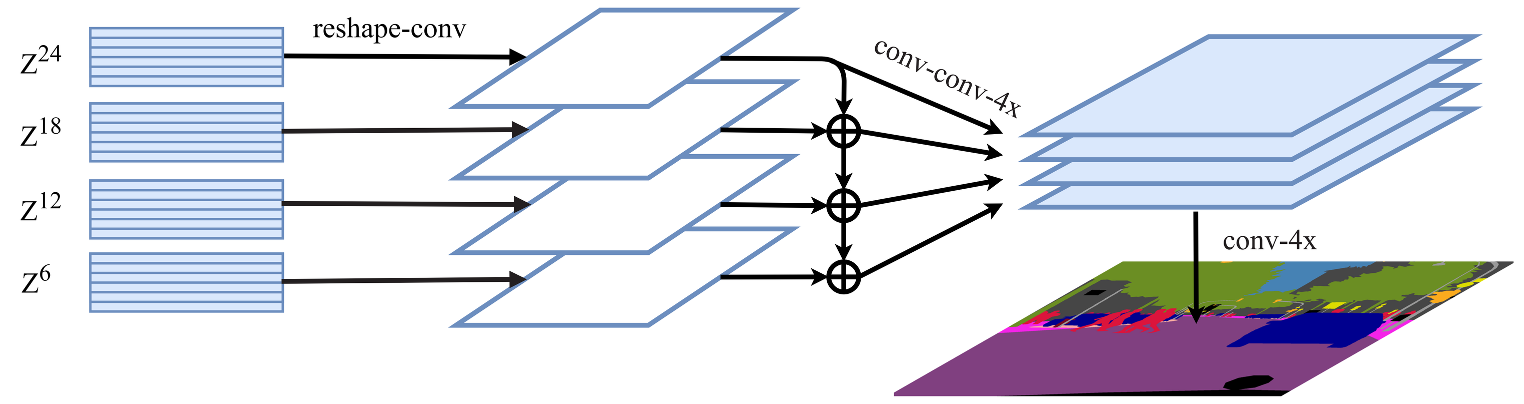

Figure 3. Multi-Level feature Aggregation

Naive upsampling (Naive): We adopt a simple 2-layer network with architecture: 1×1 conv + sync batch norm (w/ ReLU) + 1×1 conv, then simply bilinearly upsample the output to the full image resolution.

Progressive UPsampling (PUP): We adopt a progressive upsampling strategy that alternates conv layers and upsampling operations. Each time upsampling to $2\times$, a total of 4 operations are performed. As shown in Fig. 2.

Multi-Level feature Aggregation (MLA): As shown in Fig. 3. Input the features from 4 layers uniformly distributed across the layers to the decoder. Reshape the features to a 3D feature map. A 3-layer (1×1, 3×3, and 3×3) conv network is applied, and spatial resolution upscaled $4\times$. Introduce a top-down aggregation design after the first layer. An additional 3×3 conv is applied after the element-wise additioned feature. Obtain the fused feature from all the streams via channel-wise concatenation. Then bilinearly upsampled $4\times$ to the full resolution.

Qualitative results

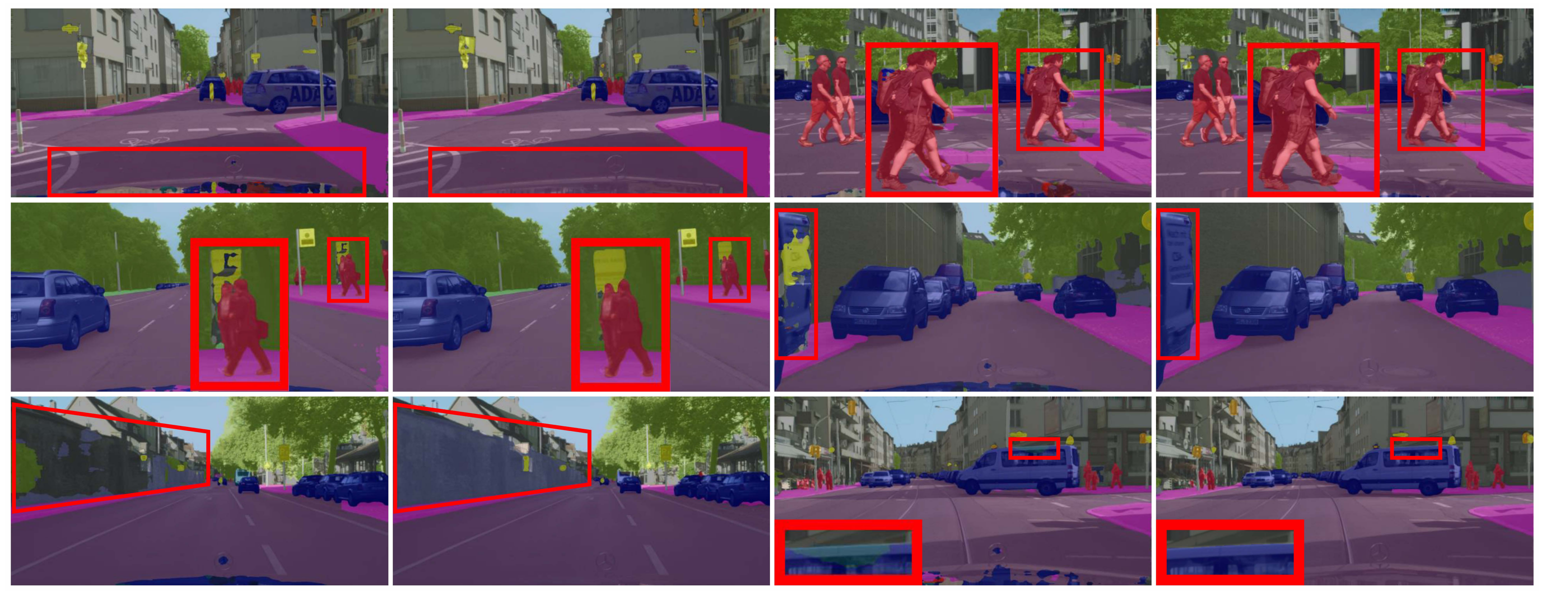

Figure 3. SETR (right column) vs. dilated FCN baseline (left column) in each pair.

Experiemts

Ablation studies:

Method	Pre	Backbone	#Params	40k	80k
FCN [38]	1K	R-101	68.59	73.93	75.52
Semantic FPN [38]	1K	R-101	47.51	-	75.80
Hybrid-Base	R	T-Base	112.59	74.48	77.36
Hybrid-Base	21K	T-Base	112.59	76.76	76.57
Hybrid-DeiT	21K	T-Base	112.59	77.42	78.28
SETR-*Naïve*	21K	T-Large	305.67	77.37	77.90
SETR-*MLA*	21K	T-Large	310.57	76.65	77.24
SETR-*PUP*	21K	T-Large	318.31	78.39	79.34
SETR-*PUP*	R	T-Large	318.31	42.27	-
SETR-*Naïve-Base*	21K	T-Base	87.69	75.54	76.25
SETR-*MLA-Base*	21K	T-Base	92.59	75.60	76.87
SETR-*PUP-Base*	21K	T-Base	97.64	76.71	78.02
SETR-*Naïve-DeiT*	1K	T-Base	87.69	77.85	78.66
SETR-*MLA-DeiT*	1K	T-Base	92.59	78.04	78.98
SETR-*PUP-DeiT*	1K	T-Base	97.64	**78.79**	**79.45**

Table 1. Comparing SETR variants.

Model	T-layers	Hidden size	Att head
T-Base	12	768	12
T-Large	24	1024	16

Table 2. Configuration of Transformer backbone variants.

Method	Pre	Backbone	ADE20K	Cityscapes
FCN [38]	1K	R-101	39.91	73.93
FCN	21K	R-101	42.17	76.38
SETR-*MLA*	21K	T-Large	**48.64**	76.65
SETR-*PUP*	21K	T-Large	48.58	78.39
SETR-*MLA-DeiT*	1K	T-Large	46.15	78.98
SETR-*PUP-DeiT*	1K	T-Large	46.24	**79.45**

Table 3. Comparison to FCN with different pre-training.

Comparison to state-of-the-art:

Method	Backbone	mIoU	Pixel Acc.
FCN (16, 160k, SS) [38]	ResNet-101	39.91	79.52
FCN (16, 160k, MS) [38]	ResNet-101	41.40	80.65
EncNet [53]	ResNet-101	44.65	81.69
PSPNet [58]	ResNet-269	44.94	81.69
DMNet [17]	ResNet-101	45.50	-
CCNet [24]	ResNet-101	45.22	-
Strip pooling [22]	ResNet-101	45.60	82.09
APCNet [18]	ResNet-101	45.38	-
OCNet [52]	ResNet-101	45.45	-
SETR-*Naïve* (16, 160k, SS)	T-Large	48.06	82.40
SETR-*Naïve* (16, 160k, MS)	T-Large	48.80	82.92
SETR-*PUP* (16, 160k, SS)	T-Large	48.58	82.90
SETR-*PUP* (16, 160k, MS)	T-Large	50.09	**83.58**
SETR-*MLA* (16, 160k, SS)	T-Large	48.64	82.64
SETR-*MLA* (16, 160k, MS)	T-Large	**50.28**	83.46

Table 4. Comparison on the ADE20K dataset.

Method	Backbone	mIoU
FCN (40k, SS) [38]	ResNet-101	73.93
FCN (40k, MS) [38]	ResNet-101	75.14
FCN (80k, SS) [38]	ResNet-101	75.52
FCN (80k, MS) [38]	ResNet-101	76.61
PSPNet [58]	ResNet-101	78.50
DeepLab-v3 [9] (MS)	ResNet-101	79.30
NonLocal [47]	ResNet-101	79.10
CCNet [24]	ResNet-101	80.20
GCNet [3]	ResNet-101	78.10
Axial-DeepLab-XL [46] (MS)	Axial-ResNet-XL	81.10
Axial-DeepLab-L [46] (MS)	Axial-ResNet-L	81.50
SETR-*PUP* (40k, SS)	T-Large	78.39
SETR-*PUP* (40k, MS)	T-Large	81.57
SETR-*PUP* (80k, SS)	T-Large	79.34
SETR-*PUP* (80k, MS)	T-Large	**82.15**

Table 5. Comparison on the Cityscapes validation set.

Method	Backbone	mIoU
FCN (16, 80k, SS) [38]	ResNet-101	44.47
FCN (16, 80k, MS) [38]	ResNet-101	45.74
PSPNet [58]	ResNet-101	47.80
DANet [16]	ResNet-101	52.60
EMANet [30]	ResNet-101	53.10
SVCNet [14]	ResNet-101	53.20
Strip pooling [22]	ResNet-101	54.50
GFFNet [29]	ResNet-101	54.20
APCNet [18]	ResNet-101	54.70
SETR-*Naïve* (16, 80k, SS)	T-Large	52.89
SETR-*Naïve* (16, 80k, MS)	T-Large	53.61
SETR-*PUP* (16, 80k, SS)	T-Large	54.40
SETR-*PUP* (16, 80k, MS)	T-Large	55.27
SETR-*MLA* (16, 80k, SS)	T-Large	54.87
SETR-*MLA* (16, 80k, MS)	T-Large	**55.83**

Table 6. Comparison on the Pascal Context dataset.

Method	Backbone	mIoU
PSPNet [58]	ResNet-101	78.40
DenseASPP [48]	DenseNet-161	80.60
BiSeNet [50]	ResNet-101	78.90
PSANet [59]	ResNet-101	80.10
DANet [16]	ResNet-101	81.50
OCNet [52]	ResNet-101	80.10
CCNet [24]	ResNet-101	81.90
Axial-DeepLab-L [46]	Axial-ResNet-L	79.50
Axial-DeepLab-XL [46]	Axial-ResNet-XL	79.90
SETR-*PUP* (100k)	T-Large	81.08
SETR-*PUP*‡	T-Large	81.64

Table 7. Comparison on the Cityscapes test set.

Visualisation

Table 4. Comparison on the ADE20K dataset.

Table 5. Comparison on the Cityscapes validation set.