Supplementary Material: Superpixel Positional Encoding to Improve ViT-based Semantic Segmentation Models

Roberto Amoroso[1]
roberto.amoroso@unimore.it
Matteo Tomei[2]
matteo.tomei@prometeia.com
Lorenzo Baraldi[1]
lorenzo.baraldi@unimore.it
Rita Cucchiara[1]
rita.cucchiara@unimore.it

[1] University of Modena and Reggio Emilia
Modena, Italy
[2] Prometeia
Bologna, Italy

1 Overview

We present a comprehensive analysis of the impact on inference time resulting from the utilization of our proposed superpixel-based positional encoding (PE). We specifically focus on two processes: data loading and model forward. Additionally, we investigate the computational overhead incurred when varying the number and compactness of superpixels, as discussed in Section 2. Our ablation studies presented in Section 3 explore alternative superpixel algorithms and the performance implications of the injection point, compactness, and number of superpixels on our superpixel-PE method. Furthermore, we provide supplementary qualitative examples in Section 4, demonstrating the efficacy of our approach across different backbones.

2 Runtime analysis

In this Section, we conduct an analysis of the impact of our proposed superpixel-PE on inference time. As described in Section 3 of the main paper, our methodology requires extracting superpixels from the input images, calculating a sinusoidal positional encoding based on superpixel centroids, and adding this PE map to the attentive features extracted by a ViT-based backbone. The first operation (superpixel extraction) impacts the data loading time, while the latter (calculating PE maps and summation) affects the forwarding time. To accomplish superpixel extraction, we adopted an optimized variant of SLIC [1], named FastSLIC[1]. For the data loading process, we fix the number of subprocesses for data loading to 8. All timings have been measured using an Nvidia(R) GeForce RTX 2080 Ti GPU and an Intel(R) Core(TM) i9-9820X CPU.

[1] https://github.com/Algy/fast-slic

Table 1: Training time with an input resolution of 480×480 for ADE20K [8] and 512×512 for Cityscapes [2], both with and without our positional encoding method. We consider 16000 superpixels with a compactness of 20.

	Data train (ms)	Training (ms)	Data eval (ms)	Inference (ms)
ADE20K:				
DPT-B [4]	1.3	151.8	1.0	32.2
DPT-B+PE$_{\mathcal{L}}$	2.2	154.9	2.1	33.9
SegFormer-B4 [7]	1.9	260.4	1.1	62.4
SegFormer-B4+PE$_{\mathcal{L}}$	2.6	262.1	1.9	66.6
Cityscapes:				
DPT-B [4]	1.3	165.3	1.2	35.3
DPT-B+PE$_{\mathcal{L}}$	4.1	168.7	4.3	38.0
SegFormer-B4 [7]	1.8	244.0	1.1	63.0
SegFormer-B4+PE$_{\mathcal{L}}$	4.7	246.6	4.1	68.2

We should remark that for our approach, execution time is a more relevant metric than FLOPs because it accounts for the overhead incurred by the data loading process, which is a consequence of the superpixel algorithm. Consequently, the use of FLOPs as a metric would not accurately reflect this overhead, albeit limited. We measure the data loading and forwarding times of the DPT-Base [4] and SegFormer-B4 [7] semantic segmentation architectures, with a single image per batch. The measured times are averaged over the ADE20K [8] validation set, comprising 2000 images cropped at a resolution of 480×480, and the Cityscapes [2] validation set, which consists of 500 images cropped to 512×512. Table 1 provides insights into data loading, training, and inference times on both ADE20K and Cityscapes. We measure the data loading time both during the training (*i.e.*, *Data train*) and inference (*i.e.*, *Data eval*). The superpixel extraction affects the data loading time considerably (about $2\times$), while the impact of superpixel-PE encoding calculation and summation is negligible and brings instead an increase of inference time under 8% (around 1 to 5 additional milliseconds per image). However, the data loading time is negligible compared to the inference time ($\sim$1 msec vs $\sim$63 msec), and training time ($\sim$2 msec vs $\sim$260 msec). Thus, our approach does not impose significant overhead on either training or inference.

Moreover, in Table 2 we report how data loading and forward times vary when changing the number of superpixels and the compactness value, given a fixed input resolution of 512×512. As expected, FastSLIC is slower as the number of superpixels increases, while its efficiency is not affected by compactness. Once again, the majority of the relative overhead is associated with the data loading process, while the forward time, which accounts for the majority of the overall time, is not significantly affected.

3 Ablation Studies

3.1 Evaluate additional superpixel algorithms

We assess the robustness of our method over alternative superpixel algorithms, namely Watershed [3] and SEEDS [6]. Both algorithms produce accurate boundaries with potentially irregular shapes compared to SLIC [1, 5]. We use the same backbone architecture and super-

Table 2: Inference time with 512×512 input resolution from Cityscapes [2], using our positional encoding method, for different values of superpixels number and compactness.

Model	#Superpixel	Compact.	Data loading (ms)	Forward (ms)
DPT-B [9]	-	-	1.2	35.3
DPT-B+PE$_{\mathcal{L}}$	4000	1	2.4	37.8
DPT-B+PE$_{\mathcal{L}}$	4000	20	2.1	38.2
DPT-B+PE$_{\mathcal{L}}$	4000	100	2.1	38.5
DPT-B+PE$_{\mathcal{L}}$	8000	1	3.0	38.5
DPT-B+PE$_{\mathcal{L}}$	8000	20	2.8	38.1
DPT-B+PE$_{\mathcal{L}}$	8000	100	3.0	38.6
DPT-B+PE$_{\mathcal{L}}$	16000	1	4.5	38.3
DPT-B+PE$_{\mathcal{L}}$	16000	20	4.3	38.0
DPT-B+PE$_{\mathcal{L}}$	16000	100	4.1	38.2

Table 3: We investigate alternative superpixel algorithms, namely Watershed [4] and SEEDS [6], on ADE20k and Cityscapes when employing a SegFormer-B0 backbone.

	FastSLIC	Watershed [4]	SEEDS [6]
ADE20K:			
SegFormer-B0+SinPE$_{\mathcal{L}}$	38.2	**38.3**	38.1
SegFormer-B0+LinearPE$_{\mathcal{L}}$	**38.4**	**38.4**	**38.4**
Cityscapes:			
SegFormer-B0+SinPE$_{\mathcal{L}}$	71.8	**72.3**	72.2
SegFormer-B0+LinearPE$_{\mathcal{L}}$	**72.2**	72.1	**72.2**

pixel number to ensure a fair comparison with our initial results. Our Superpixel-PE consistently improves segmentation performance also when integrated with Watershed and SEEDS on both the ADE20k and Cityscapes datasets, demonstrating its robustness and adaptability, as shown in Table 3. Note that Watershed and SEEDS entail a significant computational overhead compared to FastSLIC ($\times 3$ and $\times 2$ slower, respectively). Therefore, FastSLIC remains a favorable choice due to its improved performance and efficiency.

3.2 Impact of superpixel-PE injection point

In the DPT-based design, our positional encoding is resized through bilinear downsampling and added to the feature representations extracted by the encoder *before* the fusion operation applied by the decoder, as described in Section 4 of the main paper. Here we investigate the effects of varying the injection point of the superpixel-based positional encoding in encoder-decoder architectures, along with the chosen downsampling strategy. The results are summarized in Table 4. For these experiments, we consider a DPT model based on ViT-S. When switching to the Small backbone, and considering an input resolution of 224×224, we observe that a smaller number of superpixels is necessary to generate a fine-grained segments map $\mathcal{L}$, and consequently a fine-grained $PE_{\mathcal{L}}$. In this scenario, we opt for 600 superpixels and a compactness of 100. In the second row of Table 4, we demonstrate that shifting the injection of the superpixel-based positional encoding after the fusion operation but before the last convolutional layer of the decoder, does not yield beneficial results. This observation can be explained by the fact that a single convolutional layer is insufficient to capture the spatial arrangement of the centroids encoding given by the superpixel shape.

Table 4: We investigate various alternatives on where to add our positional encoding in relation to the fusion operation applied by the DPT decoder [4]. Additionally, we explore the use of a nearest downsampling strategy as an alternative to the default bilinear one.

Model	#Superpixels	Compact.	Params (M)	mIoU (%)
DPT-S [4]	-	-	37.0	38.1
DPT-S+PE$_{\mathcal{L}}$ (after fusion)	600	100	37.0	38.2
DPT-S+PE$_{\mathcal{L}}$ (nearest)	600	100	37.0	38.4
DPT-S+PE$_{\mathcal{L}}$ (before fusion)	600	100	37.0	**38.9**

Table 5: Results in terms of mean IoU on ADE20K [8] and Cityscapes [2] using SegFormer-B0 [7], for different compactness values, with and without our positional encoding. The number of superpixels is fixed to 16,000.

Model	Compact.	Params (M)	ADE20K mIoU	Cityscapes mIoU
SegFormer-B0 [7]	-	3.8	37.5	71.4
SegFormer-B0+PE$_{\mathcal{L}}$	1	3.8	37.9	71.1
SegFormer-B0+PE$_{\mathcal{L}}$	10	3.8	38.1	71.5
SegFormer-B0+PE$_{\mathcal{L}}$	20	3.8	**38.2**	**71.8**
SegFormer-B0+PE$_{\mathcal{L}}$	30	3.8	38.0	71.2

Furthermore, we also explore a nearest neighbor downsampling strategy as an alternative to bilinear interpolation for generating $PE_{\mathcal{L}}$ (to match the shape of each single f^i). However, we observe inferior performance with this approach, as indicated in the third row of Table 4.

3.3 Impact of superpixel compactness variation

In Table 5, we apply our positional encoding to the SegFormer decoder [7] and examine how variation in the superpixels compactness value impacts model performance. In this experiment, we utilize the lightweight B0 version of SegFormer with an input resolution of 512×512 and a fixed number of 16,000 superpixels. The results demonstrate that the mIoU improves as the compactness value increases from 1 to 20. However, we observe a decline in performance when the compactness is further increased to a value of 30. This can be explained by the fact that excessively increasing the compactness of the superpixels around their centroid makes them similar to polygonal patches with regular edges, invalidating the priors of the boundaries between distinct semantic classes. Qualitative evidence of this phenomenon is depicted in Figure 1 of the main paper.

3.4 Impact of superpixel number on learnable superpixel-PE

In Section 4.1 of the main paper, we introduced LearnPE, a learnable superpixel encoding approach that assigns a unique learnable vector to each superpixel, allowing the model to adaptively learn the optimal encoding during training. Building upon this, in this section, we investigate the relationship between the number of superpixels, the model size, and the mIoU performance, as presented in Table 6.

Table 6: Relationship between the number of superpixels, the model size, and the mIoU performance in the case of our learnable superpixel-based positional encoding (LearnPE$_{\mathcal{L}}$).

Model	#Superpixel	Params (M)	ADE20K mIoU	Cityscapes mIoU
SegFormer-B0 [7]	-	3.8 (+0.0)	37.5	71.4
+LearnPE$_{\mathcal{L}}$	200	3.8 (+0.0)	37.8	71.7
+LearnPE$_{\mathcal{L}}$	600	3.9 (+0.1)	38.1	71.9
+LearnPE$_{\mathcal{L}}$	4,096	4.8 (+1.0)	**38.3**	**72.2**
+LearnPE$_{\mathcal{L}}$	16,384	8.0 (+4.2)	38.0	71.7
+LearnPE$_{\mathcal{L}}$	20,000	8.9 (+5.1)	37.7	71.1
+LearnPE$_{\mathcal{L}}$	24,000	9.9 (+6.1)	37.6	71.1
+LearnPE$_{\mathcal{L}}$	32,000	12.0 (+8.2)	37.7	71.9

As the number of superpixels increases, we observe a corresponding growth in the number of parameters, surpassing three times the size of the baseline model. However, despite the increase in model capacity, the mIoU performance tends to decrease, albeit still surpassing the baseline. This suggests a potential trade-off between model complexity and performance when utilizing LearnPE. It is worth noting that while a larger model may possess more capacity, the heightened complexity may undermine the advantages of leveraging the priors provided by the superpixels, potentially leading to overfitting.

4 Additional qualitative results

Figure 1 showcases supplementary qualitative results obtained from ADE20K (first two columns) and Cityscapes (last two columns), illustrating the performance of existing Vision Transformers with and without our superpixel positional encoding strategy. Specifically, we present results for the DPT-B, SegFormer-B0, SegFormer-B4, and SETR-T models, displaying two samples for each model. Enhanced semantic segmentation outcomes are denoted by yellow boxes, indicating improved performance.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. PAMI*, 2012.

[2] 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.

[3] Fernand Meyer. Color image segmentation. In *ICIP*, 1992.

[4] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[5] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *CVIU*, 2018.

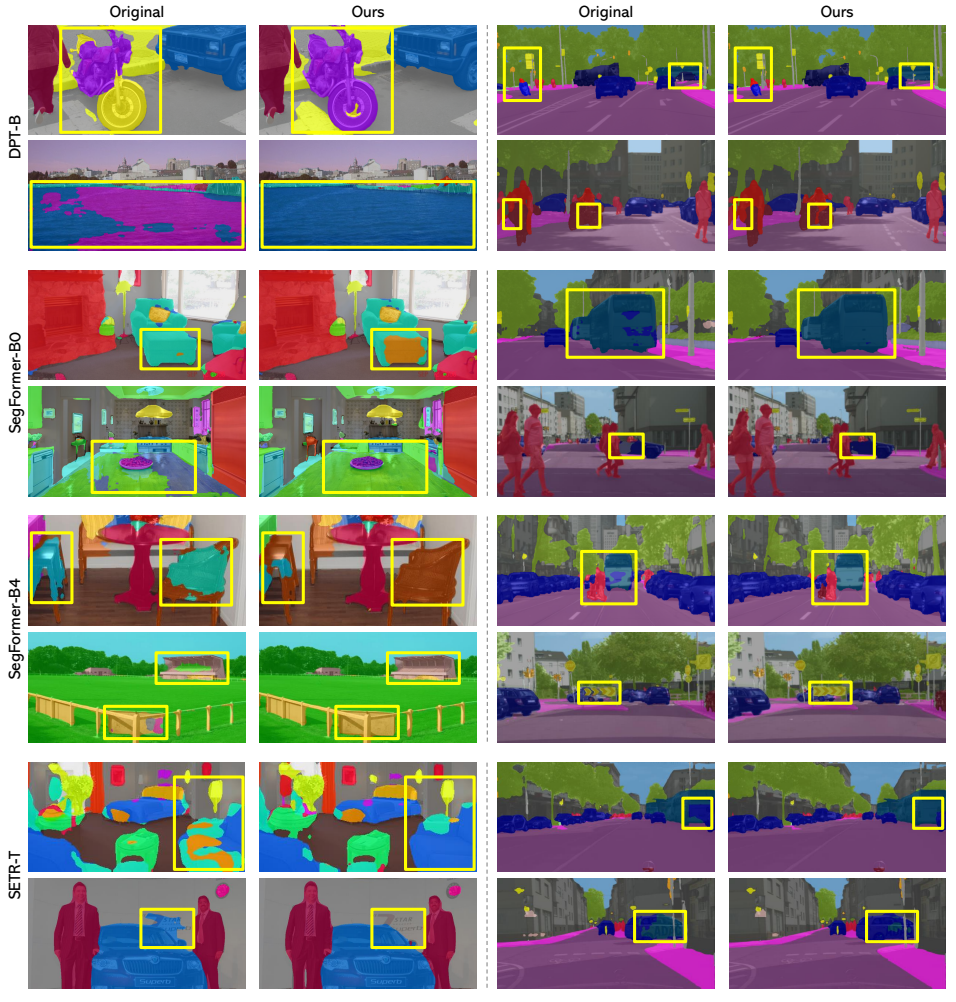

Figure 1: Sample results obtained employing DPT-B, SegFormer-B0, SegFormer-B4, and SETR-T, with and without our superpixel positional encoding strategy.

[6] Michael Van den Bergh, Xavier Boix, Gemma Roig, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *IJCV*, 2015.

[7] 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.

[8] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017.