This is the peer reviewed version of the following article:


Terms of use:
The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

08/11/2023 07:11

(Article begins on next page)
Superpixel Positional Encoding to Improve ViT-based Semantic Segmentation Models

Roberto Amoroso
roberto.amoroso@unimore.it

Matteo Tomei
matteo.tomei@prometeia.com

Lorenzo Baraldi
lorenzo.baraldi@unimore.it

Rita Cucchiara
rita.cucchiara@unimore.it

1 University of Modena and Reggio Emilia
Modena, Italy

2 Prometeia
Bologna, Italy

Abstract

In this paper, we present a novel superpixel-based positional encoding technique that combines Vision Transformer (ViT) features with superpixels priors to improve the performance of semantic segmentation architectures. Recently proposed ViT-based segmentation approaches employ a Transformer backbone and exploit self-attentive features as an input to a convolutional decoder, achieving state-of-the-art performance in dense prediction tasks. Our proposed technique is plug-and-play, model-agnostic, and operates by computing superpixels over the input image. It determines a positional encoding based on the centroids and shapes of the superpixels, and then unifies this semantic-aware information with the self-attentive features extracted by the ViT-based backbone. Our results demonstrate that this simple positional encoding strategy, when applied to the decoder of ViT-based architectures, leads to a significant improvement in performance without increasing the number of parameters and with negligible impact on the training time. We evaluate our approach on different backbones and architectures and observe a significant improvement in terms of mIoU on the ADE20K and Cityscapes datasets. Notably, our approach provides improved performance on classes with low occurrence in the dataset while mitigating overfitting on classes with higher representation, ensuring a good balance between generalization and specificity.

1 Introduction

In this work, we investigate the effectiveness of classical perception-based Computer Vision approaches when combined with more recent, self-attentive, approaches for dense tasks. The rationale behind this idea is that low-level assumptions and knowledge-based models, such as a perceptual grouping based on appearance similarity, can potentially be useful for improving the accuracy of semantic segmentation in generic images.

In recent years, the Transformer architecture [41] has received a relevant interest from the natural language processing (NLP) community [14, 36] and has established a new state-of-
the-art in fundamental vision tasks such as classification [15, 40], detection [7], and segmentation [58]. In particular, recently proposed architectures such as DPT [37], SegFormer [48], and SETR [58], have successfully integrated attentive architectures for dense prediction.

Transformer-based architectures share the common characteristic of employing a positional encoding strategy to encode the relative or absolute position of words or tokens in a sequence. This is particularly important in the context of NLP, where the order of words in a sentence profoundly impacts its meaning. Similarly, in Vision Transformers, positional encoding is used to encode the relative position of patches in an image, facilitating the understanding of spatial relationships among these objects and enabling more accurate predictions.

Although there are several works investigating positional encoding strategies for classification [15, 24, 34, 37, 47, 48], the literature on positional encoding techniques for dense prediction tasks, such as segmentation, is scarce. Moreover, existing positional encoding strategies share a common limitation, as they solely provide information about the position of input patches, while lacking semantic priors that offer insights into the shape and edges between different objects, i.e., semantic classes, within the input image.

To address this limitation, we discuss if and when a grouping strategy based on appearance similarity, such as the one of superpixels [38], can be useful for semantic segmentation when employing self-attentive-based architectures. Superpixels cluster adjacent and perceptually similar pixels in uniform image regions by following edges and color variations. As such, they have often been adopted in segmentation before the appearance of CNN-based methods. Since the seminal work by Ren and Malik [38], tens of approaches have been proposed to create superpixels according to different optimization functions [1, 29], and also Convolutional Networks have been adopted for creating superpixels [50]. Regardless of the generation approach, superpixels can be considered seed areas for segmentation tasks, as they provide a precise indication of where edges lie and have a perceptual meaning. In Transformer-based architectures, therefore, superpixels may be useful to recover precise boundaries between classes.

Building upon these insights, we devise a novel superpixel-based positional encoding (PE) strategy specifically designed for semantic segmentation. Our strategy injects superpixel shape and position priors into the ViT encoder features, creating more boundary-aware semantic latent space representations. In our experimental evaluation, we also investigate an ablation of several positional encoding strategies to encode and embed superpixel information in the encoder features.

Contributions. To sum up, our contributions are as follows:

- We investigate the integration of superpixels in ViT-based architectures for semantic segmentation.
- We propose a comprehensive comparison of several positional encoding strategies based on superpixels centroids and shape.
- We conduct experiments on the ADE20K and the Cityscapes datasets, employing different backbones and architectures. Results outline that the proposed positional embedding provides a significant improvement in terms of mIoU. Surprisingly, our approach is particularly effective in enhancing the segmentation performance for classes with low occurrence in the dataset, while also mitigating the risk of overfitting on classes with higher representation.
2 Related Work

Superpixels. Observing that pixels are not natural and meaningful entities to represent images, the first reference to superpixels as a preprocessing step can be found in [38]. Several alternatives for superpixel extraction followed, which can be classified based on their high-level approach, according to the categorization reported in [1] and [39]. Besides watershed-based algorithms [4, 32, 33], methods based on clustering techniques [1, 27, 42, 46], such as $k$-means, use pixel color, and spatial information and allow to specify the desired number of superpixels and their compactness. Other strategies instead treat images like a graph and partition their edges based on color similarities [17, 20, 29, 38]. Deep learning-based approaches have recently been proposed for superpixel computation [25, 50], and superpixels have been exploited for preserving edges and improving semantic segmentation [18, 49]. More recently, [55] applies a superpixel algorithm to the input image to extract semantically homogeneous regions and segment the image by per-region prediction using a sequence-to-sequence Transformer. For a comprehensive description and evaluation of state-of-the-art superpixel algorithms, we refer the reader to [39].

Semantic segmentation. Semantic segmentation is the dense prediction task of assigning a semantic label to each pixel in an image and can be seen as an extension of image classification at the pixel level. It is a fundamental problem in computer vision and has numerous applications, such as object detection, scene understanding, and image editing. Recently, several deep learning-based approaches have been proposed for semantic image segmentation. One of the most successful approaches is the fully convolutional network (FCN) [31], which enabled outstanding progress in semantic segmentation, performing pixel-to-pixel classification in an end-to-end manner. Many recent efforts have been focused on improving different aspects of FCN. Several works [9, 10, 11] enlarged the receptive field by adopting dilated convolutions, while others introduced context modeling [21, 28, 35, 52, 54], boundary information refinement [2, 3, 5, 8, 51] and multi-scale feature aggregation [43, 56] to obtain fine-grained predictions. Recently, attention-based models [19, 23, 26, 45, 57] have been employed to learn long-range context information. While these approaches adopt convolutional encoding backbones [22] for feature extraction, more recent methods have proved the effectiveness of employing Transformer-based [41] backbones for semantic segmentation.

Vision Transformers for dense prediction. Transformer-based architectures [41] have gained increasing attention from the computer vision community [15, 40], matching or even improving CNNs performance. Dense prediction tasks, specifically semantic segmentation, have also been tackled by adding convolutional or MLP decoders on top of ViT-based architectures. One key advantage of ViTs is their ability to process high-resolution images without downscaling, allowing the retention of more detailed information about the objects in an image and leading to improved segmentation accuracy. Ranftl et al. [37] propose to reshape tokens from different layers of a ViT-like backbone into an image representation with decreasing resolution and increasing channels. Other approaches focus on reducing memory and model parameters, by downsampling the encoder sequence length in subsequent layers [44] and proposing a lightweight MLP decoder [48], by computing self-attention only within local groups of patches [12, 30], or by transposing the query-key interaction to enable a linear complexity with respect to the number of patches [16]. More recently, Zhang et al. [53] propose a plain transformer encoder-decoder architecture that employs a novel attention-to-mask mechanism to generate segmentation masks based on the similarity between learnable class tokens and multi-level ViT feature maps.
#3 Method

For dense prediction tasks, Vision Transformers are commonly designed following an encoder-decoder architecture \[37, 48, 58\], where features coming from different encoder layers are used to predict the segmentation map. In this work, we propose a superpixel-based positional encoding that injects shape and position priors into the ViT encoder features, creating more boundary-aware semantic latent space representations. The proposed approaches are model-agnostic, parameters-free and plug-and-play; they do not involve any architectural changes to the encoder or decoder and only require the extraction of a superpixel map over the input image through an algorithm that we call \( S \).

More in detail, given the superpixel algorithm \( S \) and an input image \( z \) with shape \( H \times W \times C \), we apply \( S \) to \( z \) to obtain a superpixel map \( L \) with shape \( H \times W \), and a set of centroids \( C \) with shape \( N_s \times 2 \), where \( N_s \) represents the number of superpixels extracted by \( S \). \( L \) contains predicted superpixels, i.e., each pixel in \( L \) is represented by an integer in the range \([0, N_s - 1]\) specifying which of the \( N_s \) superpixels it belongs to, while \( C \) contains the \((x, y)\) coordinates of the centroid of each superpixel, relative to the input image.

The generation of superpixels is influenced by an important hyperparameter, namely compactness. Varying the compactness magnitude affects shape, size, and boundary smoothness. Increasing compactness results in smoother superpixels, while decreasing compactness leads to irregular shapes, similar to overfitting. Higher compactness captures spatial information better and aids information extraction. However, excessive values lead to grid-like superpixel maps. Examples are reported in Figure 1.

**Superpixel-based Positional Encoding.** The literature on ViT-based architectures \[15, 47\] highlights the importance of incorporating a positional encoding into the input embedding to achieve superior model performance. Traditionally, positional encoding conveys information about the absolute \[15, 37\] or relative \[15, 48\] position of the input patches. Recent approaches also incorporate positional encoding into intermediate layers of the architecture \[24, 34\], especially when the features extracted by the ViT-based encoder are downscaled in some way, such as through pooling \[6\]. This underscores the significance of injecting position information not only at the input but also in the intermediate processing of the model.

Our superpixel-based positional encoding (PE) not only provides information about the position of input patches but also offers insights into the shape and edges between different semantic classes within the input image. The proposed method is built upon the map of superpixels \( L \) and their centroids \( C \), as outlined in the following:

1. For a given input image \( z \), we extract a map of superpixels \( L \) along with their centroids \( C \) (colored dots in Figure 2–top-left).
2. For each superpixel \( L_i \), with \( i \in [0, N_s - 1] \), we compute the encoding of its position \( PE_i \) (colored rectangles in Figure 2–top-center) with the same shape \( d_{\text{model}} \) as the latent vector size of the Vision Transformer. By sharing the same latent shape, our Superpixel-based PE can be added to the features extracted from the ViT encoder. The superpixel position encoding \( PE_i \) can be either absolute, when employing the

![Figure 1: Example of superpixel maps with varying resolutions and compactness.](image-url)
coordinates \((x, y)\) of the corresponding centroid \(C_i\), or relative, when using the progressive index \(i\) associated with the superpixel \(L_i\), as discussed below. Given the \(i\)-th superpixel \(L_i\), we replicate its positional encoding \(PE_i\) over every pixel belonging to \(L_i\). In Figure 2–top-right, the colored small circles represent the positional encoding \(PE\) replicated over the surface of the corresponding superpixel. Following this approach, we obtain a superpixel-based positional encoding map \(PE_L\) with shape \(H \times W \times d_{model}\). At the same time, we also extract \(N_p^2\) squared regular patches (dotted grid in Figure 2–bottom-left) that we feed as input to the ViT encoder. The \(j\)th layer of the encoder outputs a feature vector \(f^j\) (colored rectangles in Figure 2–bottom-center) with shape \(N_p \times d_{model}\), after removing the \(cls\_token\) if adopted. Finally, we downsample \(PE_L\) and sum to the encoder features map \(f^j\), before feeding the result to the downstream segmentation decoder (Figure 2–bottom). These operations are executed both at training and inference time.

In other words, the pixels in \(L\) that belong to a specific superpixel are replaced with the positional encoding \(PE\) of that superpixel. This results in a full-resolution map that encodes both the shape and position of the semantically homogeneous regions (i.e., superpixels) computed by \(S\). Since the segmentation decoder usually expects features with spatial shape \(\sqrt{N_p} \times \sqrt{N_p} \times d_{model}\) as input, obtained by reshaping \(f^j\), we simply downsample the spatial resolution of \(PE_L\) in order to match \(\sqrt{N_p} \times \sqrt{N_p}\). This downsampling operation does not affect the other components of the decoder, ensuring that they remain unmodified.

**Absolute and relative position encoding strategies.** Although superpixel maps provide priors on shape and edges, the position encoding strategy remains to be defined. As a solution, we propose two approaches to inject superpixel position information: an absolute positional encoding that exploits the information of the centroids \(C\) and a relative positional encoding that leverages the progressive index associated with each superpixel of \(L\).

Inspired by [41], our absolute superpixel positional encoding adopts sine and cosine functions of different frequencies to encode \(x\) and \(y\) coordinates of the superpixel centroids:

\[
\begin{align*}
\text{SinPE}^C_{(sup,2i)} &= \sin(c_{(sup)}/10000^{2i/d_{pe}}), & \text{SinPE}^C_{(sup,2i+1)} &= \cos(c_{(sup)}/10000^{2i/d_{pe}})
\end{align*}
\]

where \(c\) represents the \(x\) or \(y\) coordinate, and \(d_{pe}\) is equal to \(d_{model}/2\). Here \textit{sup} represents
the index of the superpixel (from 0 to $N_s - 1$), and $i$ covers the $d_{pe}$ dimensions. Afterward, $SinPE^y$ and $SinPE^x$ are concatenated along the channel dimension:

$$SinPE = (SinPE^x \parallel SinPE^y)$$

with $SinPE$ having shape $N_s \times d_{model}$. After computing $SinPE$ following Equation 2, we obtain a $N_s \times d_{model}$ encoding of the $(x,y)$ centroids coordinates $C$. We now have a sinusoidal positional encoding for each superpixel extracted from the input image. Then, following the same insight as in ❸, we replicate the absolute sinusoidal positional encoding of the $i$-th centroid $SinPE_i$ over the surface of the $i$-th superpixel. Finally, we obtain $SinPE_L$, with shape $H \times W \times d_{model}$, from $L$ by replacing the pixels belonging to a specific superpixel with the sinusoidal positional encoding of its centroid.

We also propose a relative superpixel positional encoding strategy which exploits the progressive index $sup$ assigned to each superpixel by the algorithm $S$. This index is an integer in the range $[0, N_s - 1]$ that follows the row-major order, i.e., index 0 corresponds to the first superpixel in the top-left corner and index $(N_s - 1)$ corresponds to the final superpixel in the bottom-right corner. We compute a linear relative positional encoding by normalizing the superpixel indices in the interval $[0, 1]$:

$$LinearPE_{(sup)} = \frac{sup}{N_s - 1}$$

with $LinearPE$ having shape $N_s$. We replicate the linear encoding of the superpixel index over the channel size to match the latent space size of the encoder, resulting in a $LinearPE$ with shape $N_s \times d_{model}$. As done for the sinusoidal encoding, we follow the procedure described in ❸ and replicate over the surface of the $i$-th superpixel the relative linear positional encoding $LinearPE_i$ of its index, having shape $d_{model}$. In other words, we obtain $LinearPE_L$, with shape $H \times W \times d_{model}$, from $L$ by replacing the pixels belonging to a specific superpixel with the linear positional encoding of its index. For both of the proposed approaches, $PE_L$ is spatially downsampled to match the shape of $f^j$, and added to it, as depicted in Figure 2.

In our experiments, we compare the proposed superpixel-based PE with several alternative strategies for encoding position and incorporating superpixel shape and position priors into ViT encoder features, as outlined in the following section.

4 Experiments

In this section, we first present the adopted datasets, followed by the superpixel extraction algorithm, and the considered semantic segmentation architectures. Finally, we outline the results of our proposed superpixel positional encoding strategies.

Datasets. We evaluate our proposed superpixel-based techniques over two publicly available datasets: ADE20K [59] and Cityscapes [13]. ADE20K is a challenging scene-parsing dataset with 150 fine-grained semantic concepts. It comprises a training set of 20210 images, a validation set of 2000 images, and a testing set of 3352 images. On the other hand, Cityscapes is a driving dataset for semantic segmentation consisting of 5000 fine-annotated high-resolution images, split into 2975, 500, and 1525 images for training, validation, and testing respectively. It densely annotates 19 object categories in images with urban scenes.

Superpixel algorithm. As our superpixel extraction algorithm, we adopt an optimized variant of SLIC [1], named FastSLIC. To enhance efficiency, we integrate it into the data loading

1https://github.com/Algy/fast-slic
process. FastSLIC is designed with various optimization techniques, such as color quantization, integer-only arithmetic, row subsampling, and multicore parallelization. In the Supplementary, we report an in-depth analysis of the impact of superpixel extraction on data loading, training, and inference, and explore alternative superpixel algorithms.

**DPT.** Dense Prediction Transformer (DPT) consists of an encoder-decoder architecture that employs Vision Transformers for semantic segmentation. The encoder uses a stack of transformer blocks to extract feature maps at multiple stages. The decoder reconstructs image-like feature representations and progressively fuses them into the final dense prediction through residual convolutional units. Our experiments incorporate the proposed superpixel-PE by summing it to the feature representations extracted by the encoder prior to the fusion operation applied by the decoder. We consider the variant of DPT employing a ViT-Base backbone. DPT uses random horizontal flipping and random re-scaling, with a batch size of 16 and square random crops of size 480 and 512 extracted from ADE20K and Cityscapes images. We use a cross-entropy loss, AdamW optimizer, and cosine learning rate scheduler, with a learning rate set to \((0.002 \cdot \text{bs}/512)\).

**SegFormer.** The SegFormer model is based on a hierarchical Transformer encoder to extract four feature maps \((c_1, c_2, c_3, \text{ and } c_4)\) at different resolutions, ranging from the high-resolution fine-grained features of \(c_1\) to the low-resolution coarse-grained features of \(c_4\). An MLP-based decoder then fuses these features and predicts the dense segmentation mask. We downsample and sum the proposed superpixel-based PE to the encoder outputs and feed the result to the MLP decoder. SegFormer provides encoders of different sizes, from B0 to B5, and we measure the gain of our proposed approach on the B0 and B4 encoders. SegFormer uses random resizing, horizontal flipping, and cropping to 512. It employs an AdamW optimizer for 160K iterations with a batch size of 16 for B0 and 8 for B4 on ADE20K, and a batch size of 8 on Cityscapes for both B0 and B4. The learning rate is set to \(6e^{-5}\).

**SETR.** We use the multi-level feature aggregation (MLA) variant of SETR, which produces the best results on the segmentation task \([58]\). SETR-MLA (SETR to shorten) consists of an encoder-decoder ViT-based segmentation architecture. The encoder consists of a series of Transformer blocks that extract visual features at different layers. The encoder reshapes, upsamples, and fuses the visual features through a series of convolutional blocks to generate the final dense prediction. We incorporate our superpixel-PE by summing it to the reshaped visual features before these are fused by the last stack of convolutional layers. We consider two variants of SETR employing a ViT-Tiny and ViT-Small backbone respectively. SETR uses random resizing, horizontal flipping, and cropping to 512. It employs an SGD optimizer for 160K iterations with a batch size of 4 on ADE20K, and for 80K iterations with a batch size of 8 on Cityscapes, for both the Tiny and Small variants. The learning rate is set to \(1e^{-3}\) and \(2e^{-3}\) for ADE20K and Cityscapes respectively, with a momentum of 0.9.

### 4.1 Results

**Superpixel-based PE.** Table 1 presents mean IoU results for DPT with ViT-Base backbone, SegFormer with B0 and B4 backbones, and SETR with ViT-Tiny and ViT-Small backbones, with and without our absolute sinusoidal and relative linear superpixel-based positional encoding on ADE20K and Cityscapes datasets. For ADE20K, employing our sinusoidal and linear positional encoding improves mIoU from 44.9 to 45.4 and 45.8, respectively, for DPT-B, from 37.5 to 38.4 and from 49.0 to 49.3, respectively, for SegFormer-B0 and SegFormer-B4, from 35.2 to 36.3 and from 42.7 to 43.4, respectively, for SETR-T and SETR-S. For
Table 1: Results of mIoU on ADE20K and Cityscapes using DPT-Base, SegFormer-B0, SegFormer-B4, SETR-Tiny and SETR-Small with/without superpixel positional encoding.

<table>
<thead>
<tr>
<th>#Superpixel</th>
<th>Compact.</th>
<th>Params (M)</th>
<th>ADE20K (mIoU)</th>
<th>Cityscapes (mIoU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPT-B [37]</td>
<td>-</td>
<td>102.0</td>
<td>44.9</td>
<td>71.0</td>
</tr>
<tr>
<td>DPT-B+SinPE</td>
<td>16,000</td>
<td>20</td>
<td>102.0</td>
<td>45.4</td>
</tr>
<tr>
<td>DPT-B+LinearPE</td>
<td>28,000</td>
<td>10</td>
<td>102.0</td>
<td><strong>45.8</strong></td>
</tr>
<tr>
<td>SegFormer-B0 [48]</td>
<td>-</td>
<td>3.8</td>
<td>37.5</td>
<td>71.4</td>
</tr>
<tr>
<td>SegFormer-B0+SinPE</td>
<td>16,000</td>
<td>20</td>
<td>3.8</td>
<td>38.2</td>
</tr>
<tr>
<td>SegFormer-B0+LinearPE</td>
<td>28,000</td>
<td>10</td>
<td>3.8</td>
<td><strong>38.4</strong></td>
</tr>
<tr>
<td>SegFormer-B4 [48]</td>
<td>-</td>
<td>64.1</td>
<td>49.0</td>
<td>78.4</td>
</tr>
<tr>
<td>SegFormer-B4+SinPE</td>
<td>16,000</td>
<td>20</td>
<td>64.1</td>
<td><strong>49.3</strong></td>
</tr>
<tr>
<td>SegFormer-B4+LinearPE</td>
<td>28,000</td>
<td>20</td>
<td>64.1</td>
<td><strong>49.3</strong></td>
</tr>
<tr>
<td>SETR-T [58]</td>
<td>-</td>
<td>10.2</td>
<td>35.2</td>
<td>69.3</td>
</tr>
<tr>
<td>SETR-T+SinPE</td>
<td>8,192</td>
<td>10</td>
<td>10.2</td>
<td><strong>36.3</strong></td>
</tr>
<tr>
<td>SETR-T+LinearPE</td>
<td>16,384</td>
<td>10</td>
<td>10.2</td>
<td>36.1</td>
</tr>
<tr>
<td>SETR-S [58]</td>
<td>-</td>
<td>26.7</td>
<td>42.7</td>
<td>74.6</td>
</tr>
<tr>
<td>SETR-S+SinPE</td>
<td>8,192</td>
<td>10</td>
<td>26.7</td>
<td>43.0</td>
</tr>
<tr>
<td>SETR-S+LinearPE</td>
<td>8,192</td>
<td>10</td>
<td>26.7</td>
<td><strong>43.4</strong></td>
</tr>
</tbody>
</table>

Cityscapes, mean IoU improves from 71.0 to 71.7 and 72.0 for DPT-B with sinusoidal and linear encoding, respectively, from 71.4 to 72.2 and from 78.4 to 78.6 for SegFormer-B0 and SegFormer-B4, from 69.3 to 70.1 and from 74.6 to 75.0 for SETR-T and SETR-S. We observe that a large number of superpixels is necessary to ensure the fine-grained property for $L$ and Sin/Linear-PE$_L$ consequently. The obtained results show that our positional encoding strategies are effective in improving segmentation performance, as also qualitatively shown in Figure 3. See Supplementary for analysis of varying downsampling strategies and point of injection of superpixel-PE in encoder-decoder architectures.

**Pixel-based PE.** We investigate how much of the performance boost achieved by our superpixel-PE can be ascribed to positional encoding and how much to the injection of shape and edge priors provided by the superpixels. This involves adapting the sinusoidal positional encoding from Section 3 to individual pixel coordinates. The resulting Pixel-PE is added to encoder features, but it only provides a minor performance gain, compared to our superpixel-PE. Our results suggest that the shape and edge priors provided by superpixels are critical for improving segmentation performance (2nd row of Table 2).

**Patch-based PE.** We also compare with a positional encoding method based on the square shape of the patches extracted from the input image and used by the ViT encoder to generate feature representations. Instead of encoding superpixels, we encode the numeric index that progressively identifies each patch, similar to the LinearPE approach. The number of patches determines the resolution of the positional encoding mask. Patch-size values show similar performance for both datasets, even at an extreme patch size of 1. This positional encoding method lacks shape and edge priors provided by superpixels, resulting in reduced performance compared to our superpixel-based PE, as shown in Table 2 (3rd to 5th rows).

**Weighted Superpixel-PE.** We introduce the WeightedPE method, which combines encoder features and superpixel-based PE through trainable parameters. This approach helps us understand how superpixel priors relate to the different resolutions of encoder features. Instead of downsizing and summing the superpixel-based PE to the four outputs of the SegFormer
Table 2: Comparison of our sinusoidal and linear superpixel-PE with different positional encoding strategies of injecting superpixel shape and position priors in encoder features.

<table>
<thead>
<tr>
<th>#Superpixel Compact.</th>
<th>Params (M)</th>
<th>ADE20K (mIoU)</th>
<th>Cityscapes (mIoU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SegFormer-B0 [48]</td>
<td>-</td>
<td>3.8</td>
<td>37.5</td>
</tr>
<tr>
<td>+PixelPE</td>
<td>-</td>
<td>3.8</td>
<td>37.8</td>
</tr>
<tr>
<td>+PatchPE-1</td>
<td>-</td>
<td>3.8</td>
<td>37.6</td>
</tr>
<tr>
<td>+PatchPE-4</td>
<td>-</td>
<td>3.8</td>
<td>37.5</td>
</tr>
<tr>
<td>+PatchPE-16</td>
<td>-</td>
<td>3.8</td>
<td>37.7</td>
</tr>
<tr>
<td>+WeightedPE_L_16,000</td>
<td>20</td>
<td>3.8</td>
<td>37.7</td>
</tr>
<tr>
<td>+LearnablePE_L_4,096</td>
<td>10</td>
<td>4.8</td>
<td>38.3</td>
</tr>
<tr>
<td>+SinPE_L_16,000</td>
<td>20</td>
<td>3.8</td>
<td>38.2</td>
</tr>
<tr>
<td>+LinearPE_L_28,000</td>
<td>10</td>
<td>3.8</td>
<td>38.4</td>
</tr>
</tbody>
</table>

Figure 3: We present sample results obtained employing the SegFormer-B0 [48], DPT-B [37] and SETR-T [58] models, both with and without our superpixel positional encoding.

encoder as in previous experiments, WeightedPE uses a weighted sum of encoder features and superpixel-based PE. The learnable weights are four parameters, one for each feature map generated by the encoder, all initialized to 0.10. For ADE20K, the learnable weights $\beta_1$, $\beta_2$, $\beta_3$, and $\beta_4$ were found to be 0.60, 0.51, 0.19, and 0.03, while for Cityscapes, they were 0.40, 0.30, 0.12, and 0.01, respectively. These results indicate that superpixel-PE is particularly beneficial for training high-resolution feature maps that preserve edge details between semantic classes. The proposed WeightedPE approach improves performance on both datasets, as shown in the sixth row of Table 2.

Learnable Superpixel-PE. The techniques discussed so far can be used in encoder-decoder models without modifying the architecture or adding parameters, except for the four learnable weights in WeightedPE. However, to further leverage superpixel information, we propose LearnPE, which uses a learnable superpixel encoding. Each superpixel is assigned a learnable vector, allowing the model to adaptively learn the optimal encoding for each superpixel during training. LearnPE’s performance is comparable to SinPE and LinearPE, as shown in Table 2 (7th row), but with an increase in the number of model parameters. In the Supplementary we present a study demonstrating that as the number of superpixels increases, the number of parameters grows significantly, while the mIoU performance tends to decrease, suggesting a trade-off between model complexity and performance.
Impact of superpixels on semantic classes. Our Superpixel PE approach improves mIoU scores on both Cityscapes and ADE20K datasets. We analyze the effect on individual classes and find that classes with lower average pixel count, such as traffic light, flower, and sconce in ADE20K and bus, and train in Cityscapes, show consistent improvements. This suggests that our approach is effective for infrequent classes and helps avoid overfitting highly represented classes. This represents a significant advantage, as accurate segmentation of less frequent classes is often challenging due to limited data availability and class imbalance issues. We present a comparison improvement in IoU and the average number of pixels per class in Figure 4.

5 Conclusion

In this work, we investigated the effectiveness of integrating superpixels in Transformer-based architectures for semantic segmentation. We proposed a novel superpixel-based positional encoding strategy that injects superpixel shape and position priors into the ViT encoder features, creating more boundary-aware semantic latent space representations. Experimental evaluation on ADE20K and Cityscapes datasets with different architectures demonstrated significant mIoU improvement. Our approach performs well on rare classes while avoiding overfitting on common classes, ensuring a balance between generalization and specificity. Overall, our work highlights the potential of integrating classical perception-based computer vision approaches with self-attentive architectures for dense prediction tasks.

References


