FeedFormer: Revisiting Transformer Decoder for Efficient Semantic Segmentation

Jae-hun Shim[1*], **Hyunwoo Yu**[1*], **Kyeongbo Kong**[2*], **Suk-Ju Kang**[1†]

[1] Department of Electronic Engineering, Sogang University, Seoul, 04017, Republic of Korea
[2] Department of Media School, Pukyong National University, Busan, 48547, Republic of Korea
jhshim1995@sogang.ac.kr, hyunwoo137@sogang.ac.kr, kbkong@pknu.ac.kr, sjkang@sogang.ac.kr

Abstract

With the success of Vision Transformer (ViT) in image classification, its variants have yielded great success in many downstream vision tasks. Among those, the semantic segmentation task has also benefited greatly from the advance of ViT variants. However, most studies of the transformer for semantic segmentation only focus on designing efficient transformer encoders, rarely giving attention to designing the decoder. Several studies make attempts in using the transformer decoder as the segmentation decoder with class-wise learnable query. Instead, we aim to directly use the encoder features as the queries. This paper proposes the Feature Enhancing Decoder transFormer (FeedFormer) that enhances structural information using the transformer decoder. Our goal is to decode the high-level encoder features using the lowest-level encoder feature. We do this by formulating high-level features as queries, and the lowest-level feature as the key and value. This enhances the high-level features by collecting the structural information from the lowest-level feature. Additionally, we use a simple reformation trick of pushing the encoder blocks to take the place of the existing self-attention module of the decoder to improve efficiency. We show the superiority of our decoder with various light-weight transformer-based decoders on popular semantic segmentation datasets. Despite the minute computation, our model has achieved state-of-the-art performance in the performance computation trade-off. Our model FeedFormer-B0 surpasses SegFormer-B0 with 1.8% higher mIoU and 7.1% less computation on ADE20K, and 1.7% higher mIoU and 14.4% less computation on Cityscapes, respectively. Code will be released at: https://github.com/jhshim1995/FeedFormer.

Introduction

Semantic segmentation is one of the most fundamental computer vision tasks, and it is used extensively in many real-world applications, such as autonomous driving (Cordts et al. 2016; Geiger, Lenz, and Urtasun 2012) and medical diagnoses (Ma et al. 2021). However, the accurate pixel-wise prediction of the whole image is very challenging because it requires considerations for both global and local relations.

Recently, this difficulty has been largely alleviated with the emergence of transformers. Following the significant improvement by using the transformer architectures in natural language processing (NLP), (Dosovitskiy et al. 2020) first introduced Vision Transformer (ViT) which utilizes the pure transformer encoder for vision tasks. Since then, many studies have been conducted to extend ViT to the field of the semantic segmentation (Xie et al. 2021; Zheng et al. 2021; Liu et al. 2021; Wang et al. 2021; Yu and Wu 2021; Yang et al. 2022; Jain et al. 2021). In these studies, most methods focus only on improving the efficiency of the transformer encoder, but only take a simple design or borrow from the existing design for the decoder architecture. For example, SegFormer in Figure 1 (a) shows impressive performance by designing an efficient transformer encoder with a simple ALL-MLP decoder. However, it lacks consideration for designing the segmentation decoder optimized for the transformer encoder structure.

Other studies include utilizing the transformer decoder structure in vision tasks. DEtection TRansformer (Carion et al. 2020) was the first to bring the transformer encoder-decoder structure to detection and segmentation tasks. Inspired by DETR, several works expand on DETR to employ the transformer decoder in the task of semantic segmentation (Strudel et al. 2021; Bousselham et al. 2021; Cheng, Schwing, and Kirillov 2021; Cheng et al. 2022). The main difference between the feature decoder and the learnable object query decoder in Figure 1 (a) and (b), is the use of learnable query. General transformer decoders use class-wise learnable queries to extract feature information related to each class. Therefore, they either only use the highest-level feature or need an additional pixel decoder to extract the multi-scale features. Moreover, to utilize the multi-scale features, the class-wise queries have to be decoded cumulatively for each multi-scale feature. Therefore, the feature has to pass through the transformer decoder repetitively, which causes large computational complexity. *Our strategy is to directly use features as queries instead of class-wise learnable queries. In other words, we replace an ALL-MLP feature decoder with the transformer decoder, as shown in Figure 1 (c).* Then, which level of feature should be used as query and key?

In the semantic segmentation task, consideration of low-level feature is very important because the spatial detail is

[*] These authors contributed equally.
[†] Corresponding author.

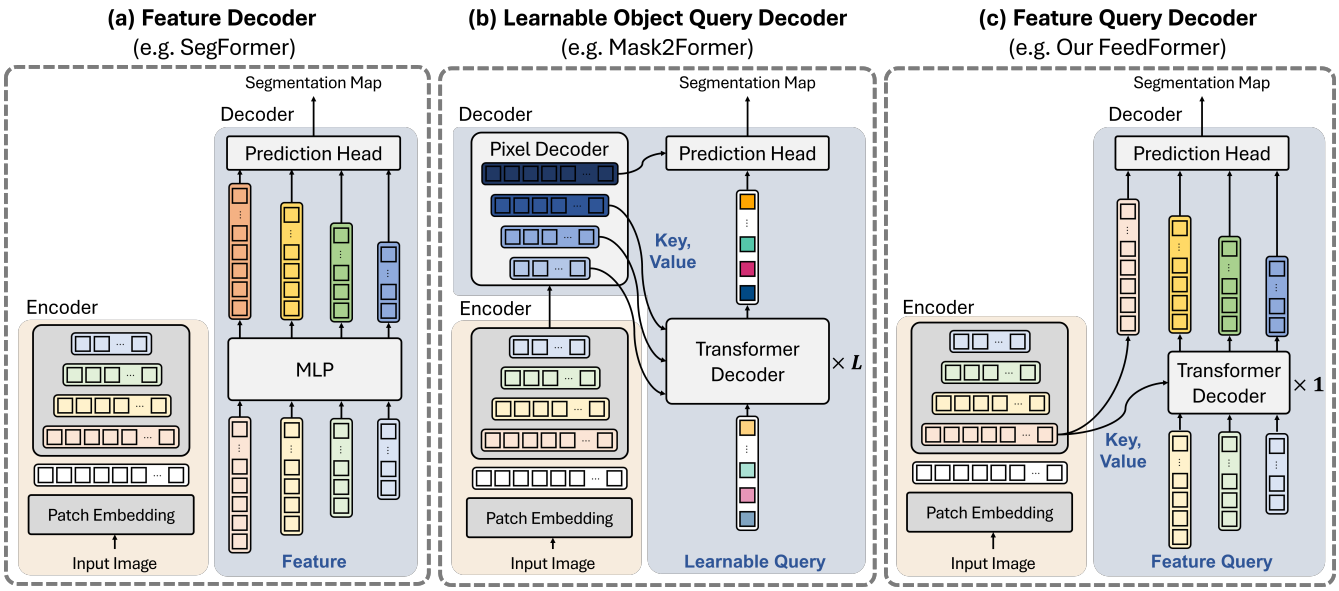

Figure 1: Comparison between the proposed and previous architectures. (a) Feature decoder (SegFormer) decodes multi-scale features from the encoder using only MLPs. (b) Architectures like Mask2Former take a learnable token as the query to use it for the mask prediction. The repetition of the transformer decoder and an additional pixel decoder is required to consider multi-scale features. (c) Our method, feature query decoder (FeedFormer), is the combination of the two methods. Our method (c) uses the *feature query* instead of the learnable query as in (b) to decode higher-level features with respect to the lowest-level feature which contains structural information.

lost as the network progresses down to high-level features. Therefore, how to pass on the low-level detail to the high-level semantic feature is of great importance. To cope with the challenge, many studies have adopted various techniques to restore lost details in the latter stages of the network (Ronneberger, Fischer, and Brox 2015; Chen et al. 2018; Wang et al. 2020). Their precise methods all differ, but they all have a unifying goal of figuring out how to add the low-level representation to the high-level features.

This paper proposes the Feature Enhancing Decoder transFormer (FeedFormer), which improves structure information of the high-level features with the transformer decoder architecture for segmentation prediction. *Our key idea is to decode the high-level encoder features using the lowest-level encoder feature*. To do this, as shown in Figure 1 (c), we replace an ALL-MLP decoder to the transformer decoder, which considers the lowest-level feature as key and value and the higher features as queries. Intuitively, this is reasonable because using the key information, the decoder collects the value information most relevant to the query and adds the relevant information back to the query. After decoding each feature with respect to the lowest-level feature, the lowest-level feature and all decoded features are concatenated through a prediction head to obtain a segmentation map.

Fortunately, the standard transformer decoder starts with a self-attention module, which has an overlapping role with the encoder's self-attention block. Therefore, to improve efficiency, we introduce a novel reformation trick, which is to simply push the self-attention block of the encoder into the decoder to take the place of the existing self-attention module as shown in Figure 2 (d). In addition, since our FeedFormer uses the transformer decoder only once, the additional computation imposed by the decoder is very small.

We demonstrate the advantages of our architecture in terms of model size, computation, and performance on two publicly available datasets. Our contributions are summarized as follows:

- We propose a novel and efficient transformer decoder architecture for semantic segmentation, which uses features as queries instead of class-wise learnable queries. In more detail, our FeedFormer decodes high-level features (queries) with respect to the lowest-level feature (key), which serves to enhance the structural information absent in the high-level features.
- To improve efficiency, we rephrase a part of the transformer encoder as the transformer decoder and push the self-attention block of the encoder into the decoder. We also exploit the transformer decoder only once for each feature, which strengthens the efficiency of our model.
- Our methodology can be easily applied to various transformer-based encoders such as Mix Transformer (MiT) (Xie et al. 2021) and Lite Vision Transformer (LVT) (Yang et al. 2022) with a simple modification.
- The superior performance of our FeedFormer is demonstrated on widely used semantic segmentation datasets in light-weight and advanced settings. Our light-weight model, FeedFormer-LVT, achieves 41.0% mIoU on ADE20K and 78.6% mIoU on Cityscapes val, respectively with only 4.6M parameters.

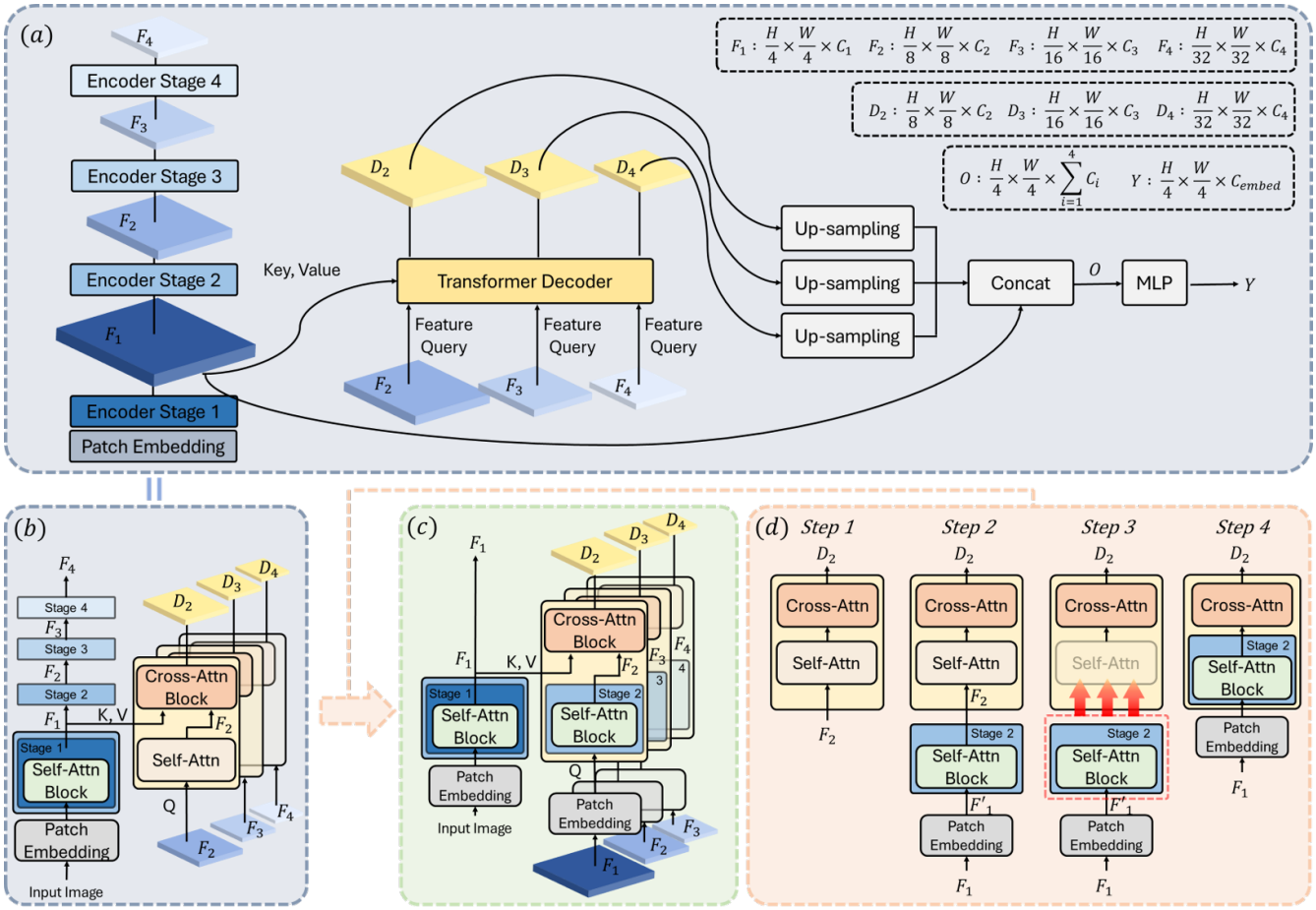

Figure 2: The proposed FeedFormer framework (a) A conceptual diagram. Among the features F_1, F_2, F_3, and F_4 extracted from the encoder, only F_1 is used as the key and value. F_2, F_3, and F_4 are used as feature queries in the transformer decoder. Through this, F_2, F_3, and F_4 are decoded by extracting the structural information related to F_1. (b) A simplified version of (a). (c) Final form of our FeedFormer architecture after applying the rearrangement trick. (d) The rearrangement process in which the self-attention layer of the transformer decoder is replaced by the encoder stage block.

Related Work

Transformer Encoder Architecture

Since the success of transformer (Vaswani et al. 2017) in NLP, the scope of transformer applications expanded to various vision tasks. ViT (Dosovitskiy et al. 2020) first proved the effectiveness of using transformer in vision task by applying it in the classification task. ViT splits an image into a sequence of patches, which is then linearly embedded as a token for the transformer block. SETR (Zheng et al. 2021) used vision transformer for the segmentation task by adopting ViT as the segmentation encoder. PVT (Wang et al. 2021) and Swin Transformer (Liu et al. 2021) tackled the problem of ViT of only being able to process single-scale features and proposed hierarchical vision transformers with four-stage design along with down-sampling features to be applied to dense prediction tasks. LVT (Yang et al. 2022) dived further into increasing the efficiency of ViT. Recently, SegFormer (Xie et al. 2021) proposed a very simple and efficient joint structure of a Mix Transformer encoder and an ALL-MLP decoder.

Transformer Decoder Architecture

Recent works adopted the transformer decoder to enhance the performance of semantic segmentation. DETR (Carion et al. 2020) was the first to apply a pure segmentation decoder for the segmentation task. DETR used the learned set of object queries to reason about the relations of the object and the context to produce a parallel set of predictions. Segmenter (Strudel et al. 2021) used a pure transformer encoder-decoder for semantic segmentation. SenFormer (Bousselham et al. 2021), MaskFormer (Cheng, Schwing, and Kirillov 2021), and Mask2Former (Cheng et al. 2022) used the transformer decoder by learning query sets, that can refine multi-scales features extracted from the backbone by stacking transformer decoders. However all these methodologies use randomly initialized learnable query, requiring an additional pixel-decoder to extract multi-scale features inputted to the transformer decoder. Our model differs in that multi-scales features are directly used from the encoder for the transformer decoder by formulating the detailed highest-level feature as the feature query.

Proposed FeedFormer Architecture

In this section, we present the proposed FeedFormer, the simple transformer encoder-decoder structure for semantic segmentation. Overall conceptual diagram is illustrated in Figure 2 (a). We first extract four multi-scale features from the hierarchical vision transformer encoder. Then, of the four multi-scale features, we use the latter three features, F_2, F_3, and F_4 as queries, and the lowest-level feature, F_1, as the key and value of the transformer decoder. This is the main difference from the existing transformer decoder, which uses learnable class-wise queries. In the decoder, pooling is performed on F_1 to match the spatial dimension of the queries. The output decoded features, D_2, D_3, and D_4, are up-sampled to match the spatial dimension of F_1. The three output decoded features, together with F_1, are concatenated and processed through the MLP layer to produce the final prediction. Our FeedFormer can also be depicted as in Figure 2 (b). Next, let us take a closer look at the inside of our FeedFormer decoder.

Building Blocks of FeedFormer

The transformer conducts attention mechanisms for the embedded patches. Here, we briefly explain the patch embedding layer and the attention block.

Patch Embedding Layer For the case of most hierarchical transformers, when an image with $H \times W \times 3$ is inputted to the network, output features of each stage take the shape of $F_i \in R^{(H_i \times W_i \times C_i)}$. Each stage feature F_i is then divided into $\frac{H \times W}{2^{i+1} \times 2^{i+1}}$ number of patches, which is linearly projected to produce embedded patches with $\frac{H}{2^{i+1}} \times \frac{W}{2^{i+1}} \times C_i$ size. Linearly embedded patches serve as an input to the multi-head attention (MHA) layer in the attention block.

Attention Block In the attention block of Figure 3, the MHA layer with query, key, and value, each denoted as **Q**, **K** and **V**, is computed as follows:

$$\mathbf{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\mathbf{Q}\mathbf{K}^{\mathrm{T}})\mathbf{V}. \quad (1)$$

This MHA layer and the feed forward layer (FFL) are sequentially connected, with layer norms (LNs) applied before every block and residual connections after every block:

$$\mathbf{a}_{j-1} = \mathbf{MHA}(\mathbf{LN}(\mathbf{kv}_{j-1}, \mathbf{q}_{j-1})) + \mathbf{q}_{j-1}, \quad (2)$$

$$\mathbf{q}_j = \mathbf{FFL}(\mathbf{LN}(\mathbf{a}_{j-1})) + \mathbf{a}_{j-1}, \quad (3)$$

where $\mathbf{q}_{j-1}$ is the embedded patches of query, $\mathbf{kv}_{j-1}$ is the embedded patches of key, value, and $j \in \{1, ..., L_i\}$. L_i is the number of repeated attention blocks.

The main difference between a self-attention layer and a cross-attention layer is how to configure the key, query, and value. The self-attention layer uses the same feature as the key, query, and value, whereas the cross-attention layer uses one feature as the query and another feature as the key and value. In our FeedFormer, the cross-attention layer regards the lowest-level feature as the key and value and high-level features as the queries. Through the attention mechanism, it collects the detail information highly relevant to the query feature from the lowest-level feature and adds it back to each high-level feature. This process has the effect of improving the detail of the high-level features regardless of the feature scales.

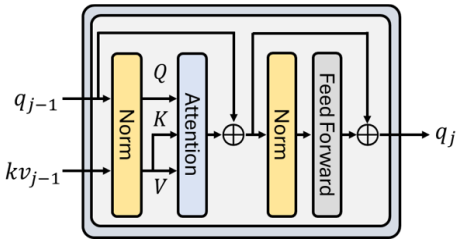

Figure 3: Attention block in a general transformer architecture. kv_{j-1} denotes the feature for key and value, and q_{j-1} denotes the query feature.

Reforming FeedFormer for Efficient Architecture

Intuitively, the feature queries of the transformer decoder are sequentially connected. Therefore, as in Steps 1 and 2 in Figure 2 (d), F_{i+1} can be represented as a function of F_i, where $Stage_i$ denotes each encoder stage, and $PatchEmbed$ denotes the Patch Embedding layer:

$$F_{i+1} = Stage_{i+1}(PatchEmbed(F_i)). \quad (4)$$

Fortunately, the typical transformer decoder starts with a self-attention module, which has an overlapping role with the encoder's self-attention block. Therefore, to enhance efficiency, we simply push the self-attention block of the encoder into the decoder to take the place of the existing self-attention module (Step 3). As a result, the query of the transformer decoder is changed to the patch embedded feature of the previous stage (Step 4). This process is equally applied to all three decoder stages. Our final architecture results in Figure 2 (c), where only the first stage is used as the encoder, and the other stages are incorporated into the decoder. Note that the feature queries are changed by the encoder's previous stage features. *To the best of our knowledge, we are the first that the part of the encoders has been reinterpreted as decoders to make an efficient structure.*

In general, the learnable query-based transformer decoders repeatedly perform decoding to utilize multi-scale features. In contrast, our FeedFormer uses rich features of encoders as queries, thus, structural information can be transferred well by only decoding each feature once.

Experiments

Experimental Settings

Datasets We conducted experiments on two publicly available datasets, ADE20K (Zhou et al. 2017) and Cityscapes (Cordts et al. 2016). ADE20K is a challenging scene parsing dataset covering 150 fine-grained semantic concepts. It consists of a training set of 20,210 images, a validation set of 2,000 images, and a testing set of 3,352 images. Cityscapes is an urban driving scene dataset for semantic segmentation consisting of 5,000 finely annotated with 19 categories. It contains 5,000 high-resolution images divided into a training set of 2,975 images, a validation set of 500 images and a testing set of 1,525 images.

	Method	Encoder	Params (M)	ADE20K		Cityscapes	
				GFLOPs	mIoU	GFLOPs	mIoU
Light-weight	FCN (Long, Shelhamer, and Darrell 2015)	MobileNetV2	9.8	39.6	19.7	317.1	61.5
	PSPNet (Zhao et al. 2017)	MobileNetV2	13.7	52.9	29.6	423.4	70.2
	DeepLabV3+ (Chen et al. 2018)	MobileNetV2	15.4	69.4	34.0	555.4	75.2
	SwiftNetRN (Orsic et al. 2019)	RseNet18	11.8	-	-	104.0	75.5
	Semantic FPN (Li et al. 2021)	ConvMLP-S	12.8	33.8	35.8	-	-
	SegFormer-B0 (Xie et al. 2021)	MiT-B0	3.8	8.4	37.4	125.5	76.2
	FeedFormer-B0 (Ours)	**MiT-B0**	**4.5**	**7.8**	**39.2**	**107.4**	**77.9**
	SegFormer-LVT (Yang et al. 2022)	LVT	3.9	10.6	39.3	140.9	77.6
	FeedFormer-LVT (Ours)	**LVT**	**4.6**	**10.0**	**41.0**	**124.6**	**78.6**
Advanced	CCNet (Huang et al. 2019)	ResNet-101	68.9	278.4	43.7	2224.8	79.5
	DeepLabV3+ (Chen et al. 2018)	ResNet101	62.7	255.1	44.1	2032.3	80.9
	OCRNet (Yuan, Chen, and Wang 2020)	HRNet-W48	70.5	164.8	45.6	1296.8	81.1
	Auto-DeepLab (Liu et al. 2019)	Auto-DeepLab-L	44.4	-	-	695.0	80.3
	Swin UperNet-T (Liu et al. 2021)	Swin-T	60.0	236.0	44.4	-	-
	SenFormer (Bousselham et al. 2021)	Swin-T	144.0	179.0	46.0	-	-
	Seg-S-Mask/16 (Strudel et al. 2021)	ViT-S	22.0	-	45.4	-	-
	SegFormer-B2 (Xie et al. 2021)	MiT-B2	27.5	62.4	46.5	717.1	81.0
	MaskFormer (Cheng, Schwing, and Kirillov 2021)	Swin-T	42.0	55.0	46.7	-	-
	Mask2Former (Cheng et al. 2022)	Swin-T	47.0	74.0	47.7	-	82.1
	FeedFormer-B2 (Ours)	**MiT-B2**	**29.1**	**42.7**	**48.0**	**522.7**	**81.5**

Table 1: Performance comparison with the state-of-the-art methods on ADE20K val and Cityscapes val. Compared to other methods, our model displays superior performance compared to models with similar or larger computational complexity.

Method	Params (M)	GFLOPs	mIoU
Segmenter-B0	4.4	-	32.5
FeedFormer-B0 (Ours)	**4.5**	**7.8**	**39.2**
Mask2Former-B0	23.0	51.8	41.1
FeedFormer-B2 (Ours)	**29.4**	**42.7**	**48.0**

Table 2: Performance comparison with learnable object query decoders. B0 and B2 each denote MiT-B0, MiT-B2.

Method	ADE20K		CityScapes	
	mIoU	mBA	mIoU	mBA
SegFormer-B0	37.4	31.2	76.2	53.6
FeedFormer-B0 (Ours)	**39.2**	**32.0**	**77.9**	**54.2**
SegFormer-B2	46.5	35.4	81.0	56.0
FeedFormer-B2 (Ours)	**48.0**	**35.8**	**81.5**	**56.1**

Table 3: Comparison of mBA on SegFormer and FeedFormer. The result shows consistent improvement in both segmentation mIoU and boundary mBA.

Implementation Details We used Mix Transformer (MiT) (Xie et al. 2021) and Lite Vision Transformer (LVT) (Yang et al. 2022) as our encoder backbone. Our model with MiT-B0, MiT-B2, and LVT encoder backbone were each named FeedFormer-B0, FeedFormer-B2, and FeedFormer-LVT. For our light-weight variants FeedFormer-B0, and FeedFormer-LVT, we used an embedding dimension of 128 for the final MLP before the prediction head, and 768 for FeedFormer-B2. For ADE20K, we keep the input resolution of the features when inputted to the decoder. For Cityscapes, we downscale all the features into half the input dimension before inputting them into the decoder to compensate for the high computational cost imposed by the high resolution of the image.

Decoder	Params (M)	GFLOPs	mIoU
Sum Head	3.5	5.6	36.5
Concat Head	3.6	7.6	37.2
Cross-attn Head	4.4	7.3	37.2
Ours	**4.5**	**7.8**	**39.2**

Table 4: Performance comparison with different decoder heads.

F_1	F_2	F_3	F_4	Params (M)	mIoU
K, V	-	-	-	4.5	39.2
-	K, V	-	-	4.4	38.8
-	-	K, V	-	4.1	38.1

Table 5: Performance comparison with different level features as the key and value. K, V each indicates key, value of the transformer decoder.

Training and Evaluation Settings We used the default setting based on the public codebase *mmsegmentation*[1]. We used 4 RTX 3090 GPUs for all training throughout the experiments. We used encoders pre-trained on ImageNet-1K dataset. During the training, we applied data augmentation using random resize from 0.5 to 2.0 ratios, random horizontal flipping, and random cropping to 512×512 pixel resolution for ADE20k and 1024×1024 pixel resolution for Cityscapes. We trained the models using AdamW optimizer for 160K iterations. We set the batch size to 16 for ADE20K and 8 for Cityscapes. We set the learning rate to an initial value of 6e-5, and then, used a polynomial learning rate decay schedule with factor 1.0 by default. For evaluation, we use ADE20K validation set and Cityscapes validation set to

[1]https://github.com/open-mmlab/mmsegmentation

Figure 4: Qualitative results on ADE20K and Cityscapes. Compared to SegFormer, our FeedFormer shows strength in accurately segmenting complex details.

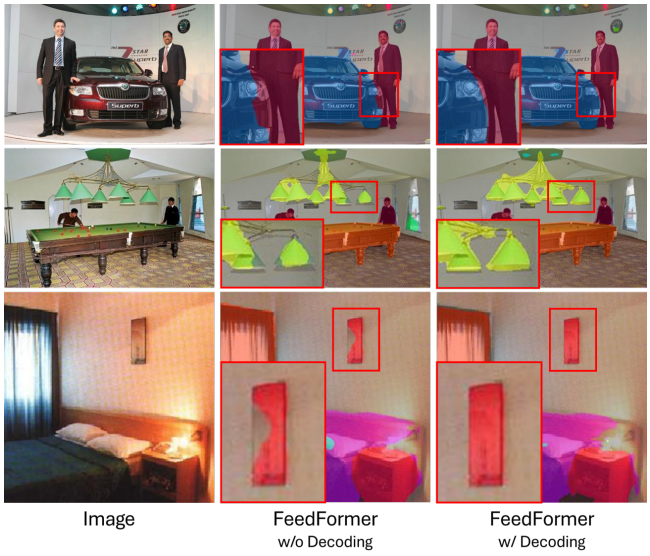

Figure 5: Visualization of ADE20K prediction results with and without the presence of transformer decoder; w/o decoding: 37.1% mIoU and w/ decoding: 38.2% mIoU.

test the performance of our model. For ADE20K, we rescaled the short side of the image to 512 and kept the aspect ratio. For Cityscapes, we applied the sliding window test by cropping 1024 $\times$ 1024 windows. We reported all our main semantic segmentation results in mean Intersection over Union (mIoU) under the single scale inference setting.

Comparison to State-of-The-Art Methods

We compared our results with the existing methods on ADE20K and Cityscapes datasets. Table 1 presents our results including parameter size, Floating Point Operations (FLOPs), and mIoU for ADE20K and Cityscapes.

Light-weight Models In the top part of Table 1, we reported the performance of the light-weight models. As shown in the table, our light-weight model FeedFormer-B0 yielded 39.2% mIoU with only 4.5M parameters and 7.8 GFLOPs for ADE20K. Compared to SegFormer-B0, FeedFormer-B0 shows 1.8% higher mIoU and 7.1% less computation. For Cityscapes, the performance gap increases even more, achieving 77.9% mIoU with only 107.4 GFLOPs. This is 1.7% increase in mIoU and 14.4% decrease in computation compared to SegFormer-B0. By using LVT as the backbone, we were able to boost the performance further, outperforming all models compared in the table. Our FeedFormer-LVT showed exceeding performance with 41.0% and 78.6% mIoU in ADE20K and Cityscapes respectively, with only 4.6M parameters.

Advanced Models In the bottom part, we show the results of the larger model, FeedFormer-B2. The results of FeedFormer-B2 showed outstanding results even in a relatively larger model. FeedFormer-B2 yielded 48.0% mIoU with only 29.1M parameters and 42.7 GFLOPs on ADE20K, which is 31.2% less computation and 1.5% better mIoU compared to SegFormer-B2. In particular, when compared to the mask2former, which uses the powerful Swin-T backbone, the number of parameters decreased by 38%, and the computation decreased by 42%, with 0.3% higher mIoU. On Cityscapes, our FeedFormer-B2 yielded 81.5% mIoU with only 522.7 GFLOPs. This is 0.5% higher mIoU and 27% less computation compared to SegFormer-B2, proving the superiority of our model on all two datasets.

Qualitative Results

Figure 4 shows the qualitative results between our FeedFormer and SegFormer on ADE20K and Cityscapes. As our method has the benefit of considering structural information when decoding multi-scale features, our model clearly shows strength in segmenting complex regions with details.

Ablation Studies

Effectiveness of Feature Decoding Figure 5 visualizes the effect of feature decoding with ADE20K prediction results. You can see that without the decoding process, inaccurate segmentation occurs in the boundaries of complex objects. Figure 6 further visualizes the effect of our detail enhancement decoder. In the figure, F_1, F_2, F_3, F_4 denote the four multi-scale features, outputted from the encoder, and D_2, D_3, D_4 denote the decoded features with respect to F_1. The figure shows that the decoded features highlight the boundary details of each multi-scale feature, which are lost before the decoding.

Effectiveness of using Feature Queries Next, to verify the effectiveness of feature decoding, we compared our FeedFormer with learnable query-based transformer decoder architecture, Segmenter (Strudel et al. 2021) and Mask2Former (Cheng et al. 2022). For a fair comparison of the decoders, we used MiT-B0 as the encoder for all experiments. As shown in Table 2, FeedFormer-B0 outperforms

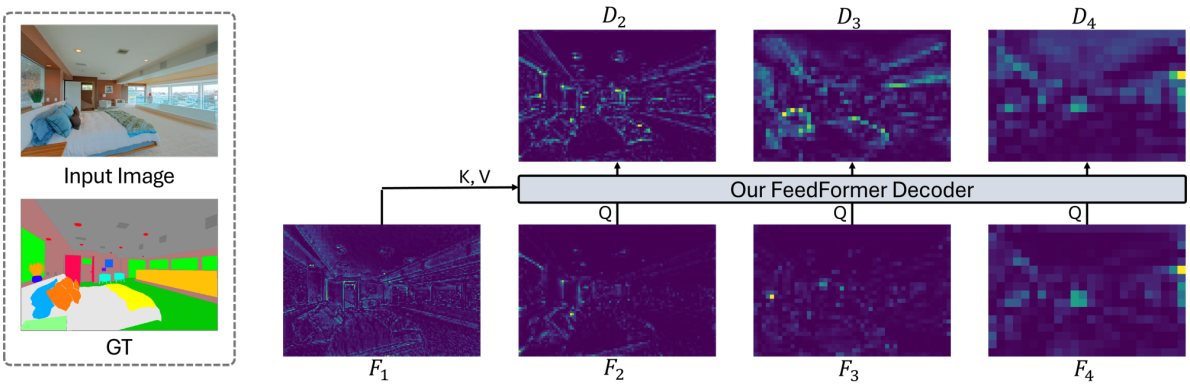

Figure 6: Visualization of feature detail enhancement of the proposed method. F_2, F_3, and F_4 denote the query features of the decoder, F_1 the key, value feature, and D_2, D_3, and D_4 denote the output of each cross-attention block. Note that boundary details from F_1 are added to F_2, F_3, and F_4 to produce output features D_2, D_3, and D_4.

Segmenter-B0, because only the highest-level feature is used from the hierarchical transformer encoder. Mask2Former-B0 shows a slight increase in performance compared to FeedFormer-B0. However, its size and computation are comparable to our larger model FeedFormer-B2, which has 6.9% higher mIoU on ADE20K. This shows that our model is more effective in decoding multi-scale features compared to transformer decoders that use learnable query.

Boundary Detail Recovery To show that using the lowest-level feature as the key and value leads to the preservation of boundary details, we evaluated our FeedFormer and SegFormer using mean Boundary Accuracy (mBA) proposed by (Cheng et al. 2020). As shown in Table 3, our FeedFormer produces segmentation predictions with strong boundaries, which result in the final segmentation performance increase.

Effectiveness of Decoding Head In this experiment, we verified the effect of our decoding head by comparing summation head (Sum Head), concatenation head (Concat Head), and cross-attention head (Cross-attn Head) to our FeedFormer head. Using the three techniques, the features were decoded with respect to the lowest-level feature F_1. For the Sum Head and the Concat Head, MLPs of embedding dimension 256 were each applied to the feature beforehand, and Cross-Attn head only used cross-attention in decoding the feature. Table 4 shows that our FeedFormer head has achieved the best performance with similar computational complexity. The result also shows that the sole use of cross-attention does not bring performance improvements. This is because the meta-architecture of transformer is essential for the performance, which is also discussed in the recent study (Yu et al. 2022). Our result shows that the same belief applies to the design of transformer decoder.

Which key, value do we choose? In Table 5, To see which feature serves the best as the key and value for the decoder, we experimented by changing the key and value of our decoder from F_1 to F_3. The results show that using the lowest-level F_1 as the key and value performs the best out of the three features. From this observation, we used the lowest-level feature F_1 as the key and value of the decoder.

Method	Params (M)	GFLOPs	mIoU	Time (ms)
PSPNet	13.7	423.4	70.2	85.5
DeepLabV3	15.4	125.5	75.2	116.3
BiSeNet	49.0	98.3	74.8	28.6
STDCNet	16.1	54.9	77.0	33.7
SegFormer	3.8	125.5	76.2	91.2
FeedFormer	**4.5**	**107.4**	**77.9**	**79.5**

Table 6: Inference time comparisons with real-time models.

Limitations Our proposed method encounters limitations when it comes to inference time. In Table 6, we represent the inference time comparisons of various models including some famous real-time segmentation models on Cityscapes dataset. We tested inference time of a single image of 2048×1024 resolution using a single RTX3090 GPU under the *mmsegmentation* benchmark without any additional accelerating techniques. The results show that our FeedFormer is slow compared to the CNN-based real-time models, BiSeNet (Yu et al. 2018) and STDCNet (Fan et al. 2021). Our model was not able to overcome the speed lag from the lack of optimization support of the transformer components, which is a general problem in many transformer-based models. We leave such limitations to future works.

Conclusion

In this paper, we present FeedFormer, a simple, pure yet powerful decoder architecture that enhances the structural information of the high-level features using the lowest-level feature. We perform this by building a transformer decoder, which takes the lowest-level feature as the key and value, and high-level features as the queries. To make our architecture more efficient, we proposed a simple reformation trick that pushes the encoder blocks to replace the existing self-attention module of the decoder. We show the superiority of our decoder with various light-weight transformer-based encoders on popular semantic segmentation datasets. For a future work, we plan to conduct experiments on various cross-attention blocks of the transformer decoder to verify our feature decoding methodology.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1004208), the National RD Program through the National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT(2021M3H2A1038042) and the Samsung Electronics Co., Ltd(IO201218-08232-01).

References

Bousselham, W.; Thibault, G.; Pagano, L.; Machireddy, A.; Gray, J.; Chang, Y. H.; and Song, X. 2021. Efficient self-ensemble framework for semantic segmentation. *arXiv preprint arXiv:2111.13280*.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, 213–229. Springer.

Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; and Adam, H. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, 801–818.

Cheng, B.; Misra, I.; Schwing, A. G.; Kirillov, A.; and Girdhar, R. 2022. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1290–1299.

Cheng, B.; Schwing, A.; and Kirillov, A. 2021. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34: 17864–17875.

Cheng, H. K.; Chung, J.; Tai, Y.-W.; and Tang, C.-K. 2020. Cascadepsp: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8890–8899.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fan, M.; Lai, S.; Huang, J.; Wei, X.; Chai, Z.; Luo, J.; and Wei, X. 2021. Rethinking BiSeNet for real-time semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9716–9725.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, 3354–3361. IEEE.

Huang, Z.; Wang, X.; Huang, L.; Huang, C.; Wei, Y.; and Liu, W. 2019. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 603–612.

Jain, J.; Singh, A.; Orlov, N.; Huang, Z.; Li, J.; Walton, S.; and Shi, H. 2021. Semask: Semantically masked transformers for semantic segmentation. *arXiv preprint arXiv:2112.12782*.

Li, J.; Hassani, A.; Walton, S.; and Shi, H. 2021. Convmlp: Hierarchical convolutional mlps for vision. *arXiv preprint arXiv:2109.04454*.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 82–92.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Ma, J.; Wang, Y.; An, X.; Ge, C.; Yu, Z.; Chen, J.; Zhu, Q.; Dong, G.; He, J.; He, Z.; et al. 2021. Toward data-efficient learning: A benchmark for COVID-19 CT lung and infection segmentation. *Medical physics*, 48(3): 1197–1210.

Orsic, M.; Kreso, I.; Bevandic, P.; and Segvic, S. 2019. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12607–12616.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Strudel, R.; Garcia, R.; Laptev, I.; and Schmid, C. 2021. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7262–7272.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, J.; Sun, K.; Cheng, T.; Jiang, B.; Deng, C.; Zhao, Y.; Liu, D.; Mu, Y.; Tan, M.; Wang, X.; et al. 2020. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10): 3349–3364.

Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 568–578.

Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J. M.; and Luo, P. 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34: 12077–12090.

Yang, C.; Wang, Y.; Zhang, J.; Zhang, H.; Wei, Z.; Lin, Z.; and Yuille, A. 2022. Lite vision transformer with enhanced self-attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11998–12008.

Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; and Sang, N. 2018. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, 325–341.

Yu, H.; and Wu, J. 2021. A unified pruning framework for vision transformers. *arXiv preprint arXiv:2111.15127*.

Yu, W.; Luo, M.; Zhou, P.; Si, C.; Zhou, Y.; Wang, X.; Feng, J.; and Yan, S. 2022. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10819–10829.

Yuan, Y.; Chen, X.; and Wang, J. 2020. Object-contextual representations for semantic segmentation. In *European conference on computer vision*, 173–190. Springer.

Zhao, H.; Shi, J.; Qi, X.; Wang, X.; and Jia, J. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2881–2890.

Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P. H.; et al. 2021. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 6881–6890.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 633–641.