Explicit Visual Prompting for Low-Level Structure Segmentations

Weihuang Liu¹ Xi Shen² Chi-Man Pun¹* Xiaodong Cun²*
¹ University of Macau ² Tencent AI Lab

Abstract

We consider the generic problem of detecting low-level structures in images, which includes segmenting the manipulated parts, identifying out-of-focus pixels, separating shadow regions, and detecting concealed objects. Whereas each such topic has been typically addressed with a domain-specific solution, we show that a unified approach performs well across all of them. We take inspiration from the widely-used pre-training and then prompt tuning protocols in NLP and propose a new visual prompting model, named Explicit Visual Prompting (EVP). Different from the previous visual prompting which is typically a dataset-level implicit embedding, our key insight is to enforce the tunable parameters focusing on the explicit visual content from each individual image, i.e., the features from frozen patch embeddings and the input’s high-frequency components. The proposed EVP significantly outperforms other parameter-efficient tuning protocols under the same amount of tunable parameters (5.7% extra trainable parameters of each task). EVP also achieves state-of-the-art performances on diverse low-level structure segmentation tasks compared to task-specific solutions. Our code is available at: https://github.com/NiFangBaAGE/Explicit-Visual-Prompt.

1. Introduction

Advances in image editing and manipulation algorithms have made it easy to create photo-realistic but fake pictures [31, 39, 63]. Detecting such manipulated regions becomes an important problem due to its potential negative impact related to surveillance and crime [31]. Low-level structures are known to be beneficial to tampered region detection, i.e., resizing and copy-pasting will destroy the JPEG compression levels between the temper region and the host image [28, 50, 62], the noise level of the tempered region and the background is also different [76, 87]. Interesting, to segment the blurred pixels [67], shadowed regions [59], and concealed objects [15], low-level clues also play important roles. These detection tasks are shown to be beneficial to numerous computer vision tasks, including auto-refocus [1], image retargeting [37], object tracking [55], etc.

Although all these tasks belong to low-level structure segmentation, they are typically addressed by domain-specific solutions with carefully designed network architectures [8, 87, 90]. Moreover, the lack of large-scale datasets is often considered a major factor, which limits the performances [31].

In this work, we propose a solution to address the four tasks in a unified fashion. We take inspiration from recent
advances of *prompting* [2, 4, 33], which is a concept that initially emerged in natural language processing (NLP) [3]. The basic idea is to efficiently adapt a frozen large foundation model to many downstream tasks with the minimum extra trainable parameters. As the foundation model has already been trained on a large-scale dataset, prompting often leads to better model generalization on the downstream tasks [3], especially in the case of the limited annotated data. Prompting also significantly saves the storage of models since it only needs to save a shared basic model and task-aware promptings.

Our main insight is to tune the task-specific knowledge *only* from the features of each individual image itself because the pre-trained base model contains sufficient knowledge for semantic understanding. This is also inspired by the effectiveness of hand-crafted image features, such as SIFT [29], JPEG noise [50], resampling artifacts [62] in these tasks [28, 29, 46, 50, 62, 87].

Based on this observation, we propose explicit visual prompting (EVP), where the tuning performance can be hugely improved via the re-modulation of image features. Specifically, we consider two kinds of features for our task. The first is the features from the frozen patch embedding, which is critical since we need to shift the distribution of the original model. Another is high-frequency components of the input image since the pre-trained visual recognition model is learned to be invariant to these features via data augmentation. As shown in Figure 1, we take a model pre-trained on a large-scale dataset and freeze its parameters. Then, to adopt to each task, we tune the embedded features and learn an extra embedding for high-frequency components of each individual image.

In terms of experiments, we validate our approach on nine datasets of four tasks: forgery detection, shadow detection, defocus blur detection as well as camouflaged object detection. Our simple and unified network achieves very competitive performance with the whole model fine-tuning and outperforms task-specific solutions without modification.

In summary, our main contributions are as follows:

- We design a unified approach that produces state-of-the-art performances for a number of tasks, including forgery detection, defocus blur detection, shadow detection, and camouflaged object detection.

- We propose explicit visual prompting (EVP), which takes the features from the frozen patch embedding and the input’s high-frequency components as prompting. It is demonstrated to be effective across different tasks and outperforms other parameter-efficient tuning methods.

- Our method greatly simplifies the low-level structure segmentation models as well as achieves comparable performance with well-designed SOTA methods.

## 2. Related Work

**Visual Prompting Tuning.** Prompting is initially proposed in NLP [3, 45]. [3] demonstrates strong generalization to downstream transfer learning tasks even in the few-shot or zero-shot settings with manually chosen prompts in GPT-3. Recently, prompting [33, 65] has been adapted to vision tasks. [65] proposes memory tokens which is a set of learnable embedding vectors for each transformer layer. VPT [33] proposes similar ideas and investigates the generality and feasibility of visual prompting via extensive experiments spanning multiple kinds of recognition tasks across multiple domains and backbone architectures. Unlike VPT, whose main focus is on recognition tasks, our work aims at exploring optimal visual content for low-level structure segmentation.

**Forgery Detection.** The goal of forgery detection is to detect pixels that are manually manipulated, such as pixels that are removed, replaced, or edited. Early approaches [7, 20, 52, 53] detect region splicing through inconsistencies in local noise levels, based on the fact that images of different origins might contain different noise characteristics introduced by the sensors or post-processing steps. Other clues are found to be helpful, such as SIFT [29], JPEG compression artifacts [50] and re-sampling artifacts [16, 62]. Recently, approaches have moved towards end-to-end deep learning methods for solving specific forensics tasks using labeled training data [26, 32, 64, 75, 86]. Salloum *et al.* [64] learn to detect splicing by training a fully convolutional network on labeled training data. [26, 47, 75, 76, 86] propose improved architectures. Islam *et al.* [32] incorporate Generative Adversarial Network (GAN) to detect copy-move forgeries. Huh *et al.* [31] propose to take photographic metadata as a free and plentiful supervisory signal for learning self-consistency and apply the trained model to detect splices. Recently, TransForensic [22] leverages vision transformers [13] to tackle the problem. High-frequency components still served as useful prior in this field. RGB-N [87] designs an additional noise stream. ObjectFormer [74] extracts high-frequency features as complementary signals to visual content. But unlike ObjectFormer, our main focus is to leverage high-frequency components as a prompting design to efficiently and effectively adapt to different low-level segmentation tasks.

**Defocus Blur Detection.** Given an image, defocus blur detection aims at separating in-focus and out-of-focus regions, which could be potentially useful for auto-refocus [1]. salient object detection [34] and image retargeting [37]. Traditional approaches mainly focus on designing hand-crafted features based on gradient [21, 67, 78] or edge [38, 68]. In the deep era, most methods delve into CNN architec-
proposes the first CNN-based method using both hand-crafted and deep features. BTB-Net [82] develops a fully convolutional network to integrate low-level clues and high-level semantic information. DeFusionNet [70] recurrently fuses and refines multi-scale deep features for defocus blur detection. CENet [83] learns multiple smaller defocus blur detectors and ensembles them to enhance diversity. [8] further employs the depth information as additional supervision and proposes a joint learning framework inspired by knowledge distillation. [79] explores deep ensemble networks for defocus blur detection. [80] proposes to learn generator to generate mask in an adversarial manner.

Shadow Detection. Shadows occur frequently in natural scenes, and have hints for scene geometry [58], light conditions [58] and camera location [36] and lead to challenging cases in many vision tasks including image segmentation [14] and object tracking [6, 56]. Early attempts explore illumination [18, 19] and hand-crafted features [30, 41, 88]. In the deep era, some methods mainly focus on the design of CNN architectures [9, 89] or involving the attention modules (e.g., the direction-aware attention [27], distraction-aware module [84]). Recent works [42, 90] utilize the lighting as additional prior, for example, ADNet [42] generates the adversarial training samples for better detection and FDRNet [90] argues the training samples by additionally adjusted brightness. MTMT [5] leverages the mean teacher model to explore unlabeled data for semi-supervised shadow detection.

Camouflaged Object Detection. Detecting camouflaged objects is a challenging task as foreground objects are often with visual similar patterns to the background. Early works distinguish the foreground and background through low-level clues such as texture [17,66], brightness [61], and color [25]. Recently, deep learning-based methods [15, 35, 44, 51, 54] show their strong ability in detecting complex camouflage objects. Le et al. [43] propose the first end-to-end network for camouflaged object detection, which is composed of a classification branch and a segmentation branch. Fan et al. [15] develops a search-identification network and the largest camouflaged object detection dataset. PFNet [54] is a bio-inspired framework that mimics the process of positioning and identification in predation. FBNet [35] suggests disentangling frequency modeling and enhancing the important frequency component.

3. Method

In this section, we propose Explicit Visual Prompting (EVP) for adapting recent Vision Transformers (SegFormer [77] as the example) pre-trained on ImageNet [10] to low-level structure segmentations. EVP keeps the backbone frozen and only contains a small number of tunable parameters to learn task-specific knowledge from the features of frozen image embeddings and high-frequency components. Below, we first present SegFormer [77] and the extraction of high-frequency components in Section 3.1, then the architecture design in Section 3.2.

3.1. Preliminaries

SegFormer [77]. SegFormer is a hierarchical transformer-based structure with a much simpler decoder for semantic segmentation. Similar to traditional CNN backbone [23], SegFormer captures multi-scale features via several stages. Differently, each stage is built via the feature embedding layers 1 and vision transformer blocks [13, 71]. As for the decoder, it leverages the multi-scale features from the encoder and MLP layers for decoding to the specific classes. Notice that, the proposed prompt strategy is not limited to SegFormer and can be easily adapted to other network structures, e.g., ViT [13] and Swin [49].

High-frequency Components (HFC). As shown in Figure 2, for an image $I$ of dimension $H \times W$, we can decompose it into low-frequency components $I_l$ (LFC) and high-frequency components $I_h$ (HFC), i.e., $I = \{I_l, I_h\}$. Denoting $\text{fft}$ and $\text{ifft}$ as the Fast Fourier Transform and its inverse respectively, we use $z$ to represent the frequency component of $I$. Therefore we have $z = \text{fft}(I)$ and $I = \text{ifft}(z)$. We shift low frequency coefficients to the center ($\frac{H}{2}, \frac{W}{2}$). To obtain HFC, a binary mask $M_h \in \{0, 1\}^{H \times W}$ is generated and applied on $z$ depending on a mask ratio $\tau$:

$$M_h^{i,j}(\tau) = \begin{cases} 0, & 4i(i-H/2)(j-W/2) \leq \tau \\ 1, & \text{otherwise} \end{cases}$$

(1)

$\tau$ indicates the surface ratio of the masked regions. HFC can be computed:

$$I_{h,fc} = \text{ifft}(zM_h(\tau))$$

(2)

1SegFormer has a different definition of patch embedding in ViT [13]. It uses the overlapped patch embedding to extract the denser features and will merge the embedding to a smaller spatial size at the beginning of each stage.
Similarly, a binary mask $M_i \in \{0, 1\}^{H \times W}$ can be properly defined to compute LFC:

$$M_i^{\tau}(\tau) = \begin{cases} 
0, & \frac{HW - 4(i-\frac{H}{2}) (j-\frac{W}{2})}{HW} \leq \tau \\
1, & \text{otherwise}
\end{cases}$$

and LFC can be calculated as:

$$I_{hfc} = \text{iff} (2M_i(\tau))$$

Note that for RGB images, we compute the above process on every channel of pixels independently.

### 3.2. Explicit Visual Prompting

In this section, we present the proposed Explicit Visual Prompting (EVP). Our key insight is to learn explicit prompts from image embeddings and high-frequency components. We learn the former to shift the distribution from the pre-train dataset to the target dataset. And the main motivation to learn the latter is that the pre-trained model is learned to be invariant to these features through data augmentation. Note that this is different from VPT [33], which learns implicit prompts. Our approach is illustrated in Figure 3, which is composed of three basic modules: patch embedding tune, high-frequency components tune as well as Adaptor.

**Patch embedding tune.** This module aims at tuning pre-trained patch embedding. In pre-trained SegFormer [77], a patch $I^p$ is projected to a $C_{seg}$-dimension feature. We freeze this projection and add a tunable linear layer $L_{pe}$ to project the original embedding into a $c$-dimension feature $F_{pe} \in \mathbb{R}^c$.

$$F_{pe} = L_{pe}(I^p), \text{with } c = \frac{C_{seg}}{r}$$

where we introduce the scale factor $r$ to control the tunable parameters.

**High-frequency components tune.** For the high-frequency components $I_{hfc}$, we learn an overlapped patch embedding similar to SegFormer [77]. Formally, $I_{hfc}$ is divided into small patches with the same patch size as SegFormer [77]. Denoting patch $I_{hfc}^p \in \mathbb{R}^C$ and $C = h \times w \times 3$, we learn a linear layer $L_{hfc}$ to project the patch into a $c$-dimension feature $F_{hfc} \in \mathbb{R}^c$.

$$F_{hfc} = L_{hfc}(I_{hfc}^p)$$

**Adaptor.** The goal of Adaptor is to efficiently and effectively perform adaptation in all the layers by considering features from the image embeddings and high-frequency components. For the $i$-th Adaptor, we take $F_{pe}$ and $F_{hfc}$ as input and obtain the prompting $P^i$:

$$P^i = \text{MLP}_{up} (\text{GELU}(\text{MLP}_{tune}^i(F_{pe} + F_{hfc})))$$

where GELU is GELU [24] activation. MLP$_{tune}^i$ is a linear layer for producing different prompts in each Adaptor. MLP$_{up}$ is an up-projection layer shared across all the Adaptors for matching the dimension of transformer features. $P^i$ is the output prompting that attaches to each transformer layer.

### 4. Experiment

#### 4.1. Datasets

We evaluate our model on a variety of datasets for four tasks: forgery detection, shadow detection, defocus blur detection, and camouflaged object detection. A summary of the basic information of these datasets is illustrated in Table 1.

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset Name</th>
<th># Train</th>
<th># Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forgery Detection</td>
<td>CAISA [12]</td>
<td>5,123</td>
<td>921</td>
</tr>
<tr>
<td>Shadow Detection</td>
<td>IMD20 [57]</td>
<td>-</td>
<td>2,010</td>
</tr>
<tr>
<td>Defocus Blur Detection</td>
<td>SBU [72]</td>
<td>1,330</td>
<td>540</td>
</tr>
<tr>
<td>Camouflaged Object Detection</td>
<td>CUHK [67]</td>
<td>4,089</td>
<td>638</td>
</tr>
<tr>
<td></td>
<td>DUT [81]</td>
<td>-</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>COD10K [15]</td>
<td>3,040</td>
<td>2,026</td>
</tr>
<tr>
<td></td>
<td>CAMO [43]</td>
<td>1,000</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>CHAMELEON [69]</td>
<td>-</td>
<td>76</td>
</tr>
</tbody>
</table>

Table 1. Summary of datasets considered in this work. We show the number of images in training (# Train) and testing set (# Test) for different datasets.

High-frequency components tune. For the high-frequency components $I_{hfc}$, we learn an overlapped patch embedding similar to SegFormer [77]. Formally, $I_{hfc}$ is divided into small patches with the same patch size as SegFormer [77]. Denoting patch $I_{hfc}^p \in \mathbb{R}^C$ and $C = h \times w \times 3$, we learn a linear layer $L_{hfc}$ to project the patch into a $c$-dimension feature $F_{hfc} \in \mathbb{R}^c$.
at the resolution of $256 \times 256$. We use pixel-level Area Under the Receiver Operating Characteristic Curve (AUC) and $F_1$ score to evaluate the performance.

**Shadow Detection.** SBU [72] is the largest annotated shadow dataset which contains 4,089 training and 638 testing samples, respectively. ISTD [73] contains triple samples for shadow detection and removal, we only use the shadowed image and shadow mask to train our method. Following [5, 89, 90], we train and test both datasets with the size of $400 \times 400$. As for the evaluation metrics, We report the balance error rate (BER).

**Defocus Blur Detection.** Following previous work [8, 81], we train the defocus blur detection model in the CUHK dataset [67], which contains a total of 704 partial defocus samples. We train the network on the 604 images split from the CUHK dataset and test in DUT [81] and the rest of the CUHK dataset. The images are resized into $320 \times 320$, with the resolution of $256 \times 256$. We use pixel-level Area Under the Receiver Operating Characteristic Curve (AUC) and $F_1$ score to evaluate the performance.

**Camouflaged Object Detection.** COD10K [15] is the largest dataset for camouflaged object detection, which contains 3,040 training and 2,026 testing samples. CHAMELEON [69] includes 76 images collected from the Internet for testing. CAMO [43] provides diverse images with naturally camouflaged objects and artificially camouflaged objects. Following [15, 54], we train on the combined dataset and test on the three datasets. We employ commonly used metrics: S-measure ($S_m$), mean E-measure ($E_\phi$), weighted F-measure ($F_\beta^w$), and MAE for evaluation.

### Table 2. Comparison with state-of-the-art approaches on defocus blur detection.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S_\alpha$ ↑</td>
<td>$E_\phi$ ↑</td>
<td>$F_\beta^w$ ↑</td>
</tr>
<tr>
<td>JCOD [44]</td>
<td>.870</td>
<td>.924</td>
<td>-</td>
</tr>
<tr>
<td>PFNet [54]</td>
<td>.882</td>
<td>.942</td>
<td>.810</td>
</tr>
<tr>
<td>FBNet [35]</td>
<td><strong>.888</strong></td>
<td>.939</td>
<td><strong>.828</strong></td>
</tr>
<tr>
<td>Ours</td>
<td>.871</td>
<td>.917</td>
<td>.795</td>
</tr>
</tbody>
</table>

### Table 3. Comparison with state-of-the-art approaches on shadow detection.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trainable Param.</th>
<th>Defocus Blur</th>
<th>Shadow</th>
<th>Forgery</th>
<th>Camouflaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-tuning</td>
<td>64.00</td>
<td><strong>.935</strong></td>
<td><strong>.039</strong></td>
<td>2.42</td>
<td>.465</td>
</tr>
<tr>
<td>Only Decoder</td>
<td>3.15</td>
<td>.891</td>
<td>.080</td>
<td>4.36</td>
<td>.396</td>
</tr>
<tr>
<td>VPT-Deep [33]</td>
<td>3.27</td>
<td>.913</td>
<td>.058</td>
<td>1.73</td>
<td><strong>.588</strong></td>
</tr>
<tr>
<td>AdaptFormer [4]</td>
<td>3.21</td>
<td>.912</td>
<td>.057</td>
<td>1.85</td>
<td><strong>.602</strong></td>
</tr>
<tr>
<td>Ours (r=16)</td>
<td>3.22</td>
<td>.924</td>
<td>.051</td>
<td>1.67</td>
<td><strong>.602</strong></td>
</tr>
<tr>
<td>Ours (r=4)</td>
<td>3.70</td>
<td>.928</td>
<td>.045</td>
<td>1.35</td>
<td><strong>.636</strong></td>
</tr>
</tbody>
</table>

### Table 4. Comparison with state-of-the-art approaches on forgery detection.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ManTra [76]</td>
<td>- .748</td>
<td>- .817</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPAN [26]</td>
<td>- .750</td>
<td>.382</td>
<td>.838</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSCCNet [48]</td>
<td>- .806</td>
<td>.554</td>
<td>.875</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransForensics [22]</td>
<td>- .848</td>
<td>627</td>
<td>.837</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ObjectFormer [74]</td>
<td>- .821</td>
<td>579</td>
<td>.882</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td><strong>.443</strong></td>
<td><strong>.807</strong></td>
<td><strong>.636</strong></td>
<td><strong>.862</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 5. Comparison with state-of-the-art approaches on camouflaged object detection.

<table>
<thead>
<tr>
<th>Method</th>
<th>Trainable Param.</th>
<th>Defocus Blur</th>
<th>Shadow</th>
<th>Forgery</th>
<th>Camouflaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-tuning</td>
<td>64.00</td>
<td><strong>.935</strong></td>
<td><strong>.039</strong></td>
<td>2.42</td>
<td>.465</td>
</tr>
<tr>
<td>Only Decoder</td>
<td>3.15</td>
<td>.891</td>
<td>.080</td>
<td>4.36</td>
<td>.396</td>
</tr>
<tr>
<td>VPT-Deep [33]</td>
<td>3.27</td>
<td>.913</td>
<td>.058</td>
<td>1.73</td>
<td><strong>.588</strong></td>
</tr>
<tr>
<td>AdaptFormer [4]</td>
<td>3.21</td>
<td>.912</td>
<td>.057</td>
<td>1.85</td>
<td><strong>.602</strong></td>
</tr>
<tr>
<td>Ours (r=16)</td>
<td>3.22</td>
<td>.924</td>
<td>.051</td>
<td>1.67</td>
<td><strong>.602</strong></td>
</tr>
<tr>
<td>Ours (r=4)</td>
<td>3.70</td>
<td>.928</td>
<td>.045</td>
<td>1.35</td>
<td><strong>.636</strong></td>
</tr>
</tbody>
</table>

### Table 6. Comparison with state-of-the-art efficient tuning approaches. We conduct evaluations on four datasets for four different tasks. The efficient tuning method which achieves better performance than full-tuning is marked as orange. The best performance among all methods is shown as **bold.**

- DUT [81]
- CAMO [43]
- SBU [72]
- ISTD [73]
- COD10K [15]
Figure 4. Comparisons with other task-specific methods. We show the results of: SINet [15] and PFNet [54] on CAMO [43] dataset for camouflaged object detection (Top-left), ManTra [76] and SPAN [26] on CAISA [12] dataset for forgery detection (Top-right), MTMT [5] and FDRNet [90] on ISTD [73] dataset for shadow detection (Bottom-left), CENet [83] and EFENet [79] on CUHK [67] dataset for defocus blur detection (Bottom-right).

Table 7. Ablation on the architecture designs described in Figure 3. We conduct evaluations on four datasets for four different tasks. The proposed prompting strategy (Decoder + $F_{pe} + F_{hfc} +$ Adaptor) performs more effectively.

4.2. Implementation Details

All the experiments are performed on a single NVIDIA Titan V GPU with 12G memory. AdamW [40] optimizer is used for all the experiments. The initial learning rate is set to $2e^{-4}$ for defocus blur detection and camouflaged object detection, and $5e^{-4}$ for others. Cosine decay is applied to the learning rate. The models are trained for 20 epochs for the SBU [72] dataset and camouflaged combined dataset [15,69], and 50 epochs for others. Random horizontal flipping is applied during training for data augmentation. The mini-batch is equal to 4. Binary cross-entropy (BCE) loss is used for defocus blur detection and forgery detection, balanced BCE loss is used for shadow detection, and BCE loss and IOU loss are used for camouflaged object detection. All the experiments are conducted with SegFormer-B4 [77] pre-trained on the ImageNet-1k [11] dataset.

4.3. Main Results

Comparison with the task-specific methods. EVP performs well when compared with task-specific methods. We report the comparison of our methods and other task-specific methods in Table 2, Table 3, Table 4, and Table 5. Thanks to our stronger backbone and prompting strategy, EVP achieves the best performance in 5 datasets across 4 different tasks. However, compared with other well-designed domain-specific methods, EVP only introduces a small number of tunable parameters with the frozen backbone and obtains non-trivial performance. We also show some visual comparisons with other methods for each task individually in Figure 4. We can see the proposed method predicts more accurate masks compared to other approaches.
We first compare EVP (equation 5) which is used to control the number of parameters of the Adaptor as described in equation 5. We introduce a hyper-parameter \( r \) which is used to control the number of parameters of the Adaptor. The number of prompt tokens is set to 10 for VPT and the middle dimension of AdaptMLP is set to 2 for a fair comparison in terms of the tunable parameters. It can be seen from Table 6 that when only tuning the decoder, the performance drops largely. Compared with similar methods, introducing extra learnable tokens [33] or MLPS in Transformer block [4] also benefits the performance. We try to answer the question: which stage contributes mostly to prompting tuning? Thus, we show the variants of our tuning method by changing the tunable stages in the SegFormer backbone. SegFormer contains 4 stages for multi-scale feature extraction. We mark the Stage \( x \) where the tunable prompting is added in Stage \( x \). Table 8 shows that better performance can be obtained via the tunable stages increasing. Besides, the maximum improvement occurs in Stage\( _{1,2} \) to Stage\( _{1,2,3} \). Note that the number of transformer blocks of each stage in SegFormer-B4 is 3, 8, 27, and 3, respectively. Thus, the effect of EVP is positively correlated to the number of the prompted transformer blocks.

**Comparison with the efficient tuning methods.** We evaluate our method with full finetuning and only tuning the decoder, which are the widely-used strategies for downstreaming task adaption. And similar methods from image classification, i.e., VPT [33] and AdaptFormer [4]. The number of prompt tokens is set to 10 for VPT and the middle dimension of AdaptMLP is set to 2 for a fair comparison in terms of the tunable parameters. It can be seen from Table 6 that when only tuning the decoder, the performance drops largely. Compared with similar methods, introducing extra learnable tokens [33] or MLPS in Transformer block [4] also benefits the performance. We introduce a hyper-parameter \( r \) which is used to control the number of parameters of the Adaptor as described in equation 5. We first compare EVP \( (r=16) \) with similar parameters as other methods. From the table, our method achieves much better performance. We also report EVP \( (r=4) \), with more parameters, the performance can be further improved and outperforms full-tuning on 3 of 4 datasets.

**4.4. Ablation Study**

We conduct the ablation to show the effectiveness of each component. The experiments are performed with the scaling factor \( r = 4 \) except specified.

**Architecture Design.** To verify the effectiveness of the proposed visual prompting architecture, we modify it into different variants. As shown in Table 7 and Figure 5, sharing MLP\(_{\text{tune}}\) in different Adaptors only saves a small number of parameters (0.55M v.s. 0.34M) but leads to a significant performance drop. It cannot obtain consistent performance improvement when using different MLP\(_{\text{tune}}\) in different Adaptors, moreover introducing a large number of parameters (0.55M v.s. 1.39M). On the other hand, the performance will drop when we remove \( F_{\text{pe}} \) or \( F_{hfc} \), which means that they are both effective visual prompts.

**Tuning Stage.** We try to answer the question: which stage contributes mostly to prompting tuning? Thus, we show the variants of our tuning method by changing the tunable stages in the SegFormer backbone. SegFormer contains 4 stages for multi-scale feature extraction. We mark the Stage \( x \) where the tunable prompting is added in Stage \( x \). Table 8 shows that better performance can be obtained via the tunable stages increasing. Besides, the maximum improvement occurs in Stage\( _{1,2} \) to Stage\( _{1,2,3} \). Note that the number of transformer blocks of each stage in SegFormer-B4 is 3, 8, 27, and 3, respectively. Thus, the effect of EVP is positively correlated to the number of the prompted transformer blocks.

**Scale Factor \( r \) (equation 5).** We introduce \( r \) in Sec 3.2 of the main paper to control the number of learnable parameters. A larger \( r \) will use fewer parameters for tuning. As shown in Table 9, the performance improves on several tasks when \( r \) increases.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$F_R \uparrow$</td>
<td>BER $\downarrow$</td>
<td>$F_1 \uparrow$</td>
<td>$S_\alpha \uparrow$</td>
</tr>
<tr>
<td>Full-tuning</td>
<td>98.98</td>
<td>.862</td>
<td>.677</td>
<td>.4.39</td>
<td>.290</td>
</tr>
<tr>
<td>Only Decoder</td>
<td>13.00</td>
<td>.836</td>
<td>.097</td>
<td>4.77</td>
<td>.318</td>
</tr>
<tr>
<td>VPT [33]</td>
<td>13.09</td>
<td>.843</td>
<td>.092</td>
<td>4.56</td>
<td>.315</td>
</tr>
<tr>
<td>EVP</td>
<td>13.06</td>
<td>.850</td>
<td>.087</td>
<td>4.36</td>
<td>.324</td>
</tr>
</tbody>
</table>

Table 10. Comparison with other tuning methods with SETR [85] on four different tasks. We conduct an evaluation on four datasets for four different tasks. The best performance is shown as **bold**. The prompt-tuning method which achieves better performance than full-tuning is marked as orange.

Figure 5. Quantitative comparison using full-tuning and different prompting designs on ISTD [73] dataset for shadow detection. From the left to right is: (a) Input, (b) GT, (c) Full-tuning, (d) Decoder (No prompting), (e) Ours w/o $F_{pe}$, (f) Ours w/o $F_{h,fc}$, (g) Ours w/ Shared MLP$_{\text{tune}}$, (h) Ours w/ Unshared MLP$_{\text{up}}$, (i) Ours Full.

decreases from 64 to 4; when $r$ continues to decrease to 2 or 1, it can not gain better performance consistently even if the model becomes larger. It indicates that $r = 4$ is a reasonable choice to make a trade-off between the performance and model size.

**EVP in Plain ViT.** We experiment on SETR [85] to confirm the generalizability of EVP. SETR employs plain ViT as the backbone and a progressive upsampling ConvNet as the decoder, while SegFormer has a hierarchical backbone with 4 stages. Therefore, the only distinction between the experiments using SegFormer is that all modifications are limited to the single stage in plain ViT. The experiments are conducted with ViT-Base [13] pretrained on the ImageNet-21k [11] dataset. The number of prompt tokens is set to 10 for VPT, the middle dimension of AdaptMLP is set to 4 for AdaptFormer, and $r$ is set to 32 for our EVP. As shown in Table 10, EVP also outperforms other tuning methods when using plain ViT as the backbone.

5. Conclusion

In this paper, we present explicit visual prompting to unify the solutions of low-level structure segmentations. We mainly focus on two kinds of features: the frozen features from patch embedding and the high-frequency components from the original image. Equipped with our method, we find that a frozen vision transformer backbone from the ImageNet with limited tunable parameters can achieve similar performance as the full-tuned network structures, also the state-of-the-art performance compared with the other task-specific methods. For future works, we will extend our approach to other related problems and hope it can promote further exploration of visual prompting.

**Acknowledgments.** This work was supported in part by the University of Macau under Grant MYRG2022-00190-FST and in part by the Science and Technology Development Fund, Macau SAR, under Grants: 0034/2019/AMJ, 0087/2020/A2, and 0049/2021/A.
References


Ali Karaali and Claudio Rosito Jung. Image retargeting based on spatially varying defocus blur map. In International Conference on Image Processing (ICIP), 2016. 1, 2


Bahjat Kawar, Shriran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. arXiv, 2022. 1


Trung-Nghia Le, Tam V Nguyen, Zhongliang Nie, Minh-Triet Tran, and Akihiro Sugimoto. Anabanch network for camouflaged object segmentation. Computer Vision and Image Understanding, 184:45–56, 2019. 3, 4, 5, 6, 7, 8


Qingzhong Liu. Detection of misaligned cropping and re-compression with the same quantization matrix and relevant forgery. In ACM workshop on Multimedia in forensics and intelligence, 2011. 2


Siwei Lyu, Xunyu Pan, and Xing Zhang. Exposing region splicing forgeries with blind local noise estimation. 2014. 2


Ivana Mikic, Pamela C Cosman, Greg T Kogut, and Mohan M Trivedi. Moving shadow and object detection in traffic scenes. 2000. 1

Sohail Nadimi and Bir Bhanu. Physical models for moving shadow and object detection in video. Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2004. 3


Takahiro Okabe, Imari Sato, and Yoichi Sato. Attached shadow coding: Estimating surface normals from shadows
under unknown reflectance and lighting conditions. In Proceedings of the International Conference on Computer Vision (ICCV), 2009. 3


[78] Xin Yi and Mark Eramian. Lbp-based segmentation of defocus blur. Transactions on Image Processing (TIP), 2016. 2


