A. Implementation Details

We give more implementation details in the main paper of the "comparison with the task-specific methods" and "comparison with the efficient tuning methods".

Basic Setting. Our method contains a backbone for feature extraction and a decoder for segmentation prediction. We initialize the weight of the backbone via ImageNet classification pre-training, and the weight of the decoder is randomly initialized. Below, we give the details of each variant.

Full-tuning. We follow the basic setting above, and then, fine-tune all the parameters of the encoder and decoder.

Only Decoder. We follow the basic setting above, and then, fine-tune the parameters in the decoder only.

VPT [33]. We first initialize the model following the basic setting. Then, we concatenate the prompt embeddings in each transformer block of the backbone only. Notice that, their prompt embeddings are implicitly shared across the whole dataset. We follow their original paper and optimize the parameters in the prompt embeddings and the decoder.

AdaptFormer [4]. We first initialize the model following the basic setting above. Then, the AdaptMLP is added to each transformer block of the backbone for feature adaptation. We fine-tune the parameters in the decoder and the newly introduced AdaptMLP.

EVP (Ours). We also initialize the weight following the basic setting. Then, we add the explicit prompting as described in the main paper of Figure 3.

Metric. AUC calculates the area of the ROC curve. ROC curve is a function of true positive rate ($\frac{tp}{tp+fn}$) in terms of false positive rate ($\frac{fp}{fp+tn}$), where tp, tn, fp, fn represent the number of pixels which are classified as true positive, true negative, false positive, and false negative, respectively. F_1 score is defined as $F_1 = \frac{2\times precision \times recall}{precision+recall}$, where $precision = \frac{tp}{tp+fp}$ and $recall = \frac{tp}{tp+fn}$. The balance error rate (BER) $= \left(1 - \frac{1}{2}\left(\frac{tp}{tp+fn} + \frac{tn}{tn+fp}\right)\right) \times 100$. F-measure is calculated as $F_\beta = \frac{\left(1+\beta^2\right)\times precision \times recall}{\beta^2 \times precision + recall}$, where $\beta^2 = 0.3$. MAE computes pixel-wise average distance. Weighted F-measure (F_β^w) weighting the quantities TP, TN, FP, and FN according to the errors at their location and their neighborhood information: $F_\beta^w = \frac{\left(1+\beta^2\right)\times precision^w \times recall^w}{\beta^2 \times precision^w + recall^w}$. E-measure ($E_\phi$) jointly considers image statistics and local pixel matching: $E_\phi = \frac{1}{W\times H}\sum_{i=1}^{W}\sum_{j=1}^{H}\phi_S(i,j)$, where ϕ_S is the alignment matrix depending on the similarity of the prediction and ground truth.

Training Data. Note that most forgery detection methods (ManTraNet [76], SPAN [26], PSCCNet [48], and ObjectFormer [74] in Table 4) and one shadow detection method (MTMT [5] in Table 3) use extra training data to get better performance. We only use the training data from the standard datasets and obtain SOTA performance.

B. More Results

We provide more experimental results in addition to the main paper.

B.1. High-Frequency Prompting

Our method gets the knowledge from the explicit content of the image itself, hence we also discuss other similar explicit clues of images as the prompts. Specifically, we choose the common-used Gaussian filter, the noise-filter [20], the all-zero image, and the original image as experiments. From Table B11, we find the Gaussian filter shows a better performance in defocus blur since it is also a kind of blur. Also, the noise filter [20] from forgery detection also boosts the performance. Interestingly, we find that simply replacing the original image with an all-zero image also boosts the performance, since it can also be considered as a kind of implicitly learned embeddings across the full dataset as in VPT [33]. Differently, the high-frequency components of the image achieve consistent performance improvement to other methods on these several benchmarks.

B.2. HFC *v.s.* LFC

We conduct the ablation study on choosing of high-frequency features or the low-frequency features in Table B12. From the table, using the low-frequency components as the prompting just show some trivial improvement on these datasets. Differently, the high-frequency components are more general solutions and show a much better performance in shadow detection, forgery detection, and camouflaged detection. Similar to the Gaussian filter as we discussed above, the LFC is also a kind of blur, which makes the advantage of LFC in the defocus blur detection.

B.3. Mask Ratio τ

We further evaluate the hyper-parameter mask ratio τ introduced in Section 3.1. From Table B12, when we mask out 25% of the central pixels in the spectrum, it achieves consistently better performance in all the tasks. We also find that the performance may drop when the increasing of mask ratio (all 0 images), especially in shadow detection, forgery detection, and camouflaged object detection.

Method	**Defocus Blur** CUHK [67]		**Shadow** ISTD [73]	**Forgery** CASIA [12]		**Camouflaged** CAMO [43]	
	$F_\beta \uparrow$	MAE $\downarrow$	BER $\downarrow$	$F_1 \uparrow$	AUC $\uparrow$	$S_\alpha \uparrow$	$E_\phi \uparrow$
Gaussian Blur	**.928**	.046	1.52	.631	.860	.842	.893
Noise Filter	.923	.046	1.47	**.637**	**.866**	.843	.894
All 0 Image	.922	.047	1.45	.630	.862	.844	.895
Original	.922	.047	1.59	.630	.860	.840	.891
HFC	**.928**	**.045**	**1.35**	.636	.862	**.846**	**.895**

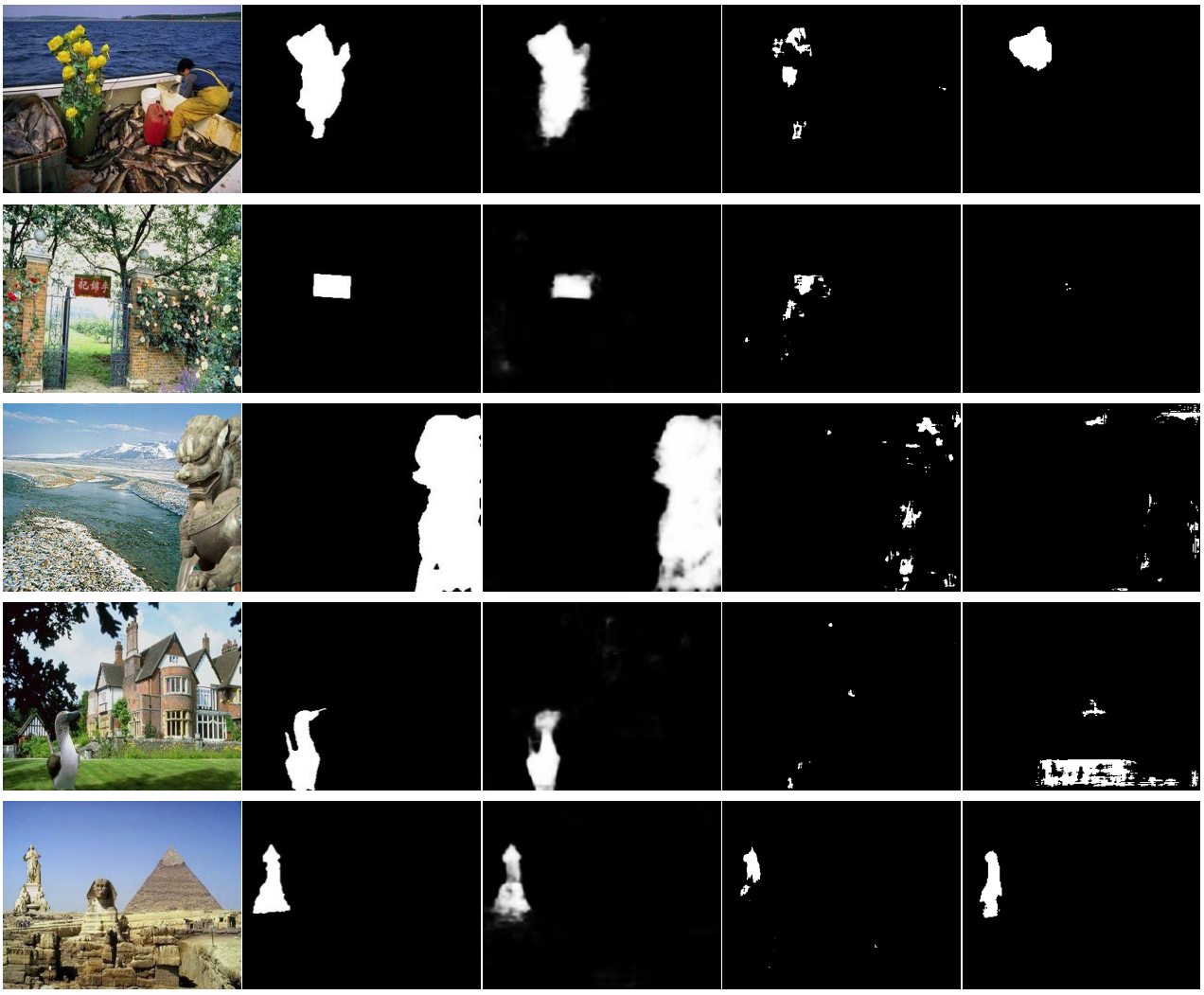
Table B11. Ablation on the explicit visual prompting with other relatives. We compare with the widely-used image filter as prompting to verify the effectiveness of the proposed HFC.

Method	Mask Ratio τ (%)	**Defocus Blur** CUHK [67]		**Shadow** ISTD [73]	**Forgery** CASIA [12]		**Camouflaged** CAMO [43]	
		$F_\beta \uparrow$	MAE $\downarrow$	BER $\downarrow$	$F_1 \uparrow$	AUC $\uparrow$	$S_\alpha \uparrow$	$E_\phi \uparrow$
Low Frequency Components (LFC) with FFT								
LFC*	0	.922	.047	1.45	.630	.862	.844	.895
LFC	10	.927	.046	1.58	.631	.862	.845	**.895**
LFC	25	.924	.047	1.49	.630	.860	.842	.891
LFC	50	.923	.048	1.48	.630	.860	.841	.893
LFC	75	.924	048	1.56	.627	.859	.840	.894
LFC	90	.925	.046	1.47	.626	.859	.841	.894
LFC**	100	.922	.047	1.59	.630	.860	.840	.891
High Frequency Components (HFC) with FFT								
HFC	10	.926	.046	1.60	.631	.854	.843	.894
HFC	25	**.928**	**.045**	**1.35**	**.636**	**.862**	**.846**	**.895**
HFC	50	.925	.047	1.51	.631	.862	.843	.894
HFC	75	.923	.048	1.52	.629	.858	.842	.892
HFC	90	.924	.047	1.49	.630	.861	.842	.893

Table B12. Ablation on HFC, LFC, and mask ratio τ. Leveraging FFT to extract high-frequency components consistently outperforms LFC. *LFC (τ=0) equals to a full zero image for prompting. **LFC (τ=100) means we learn an embedding from the original input image directly.

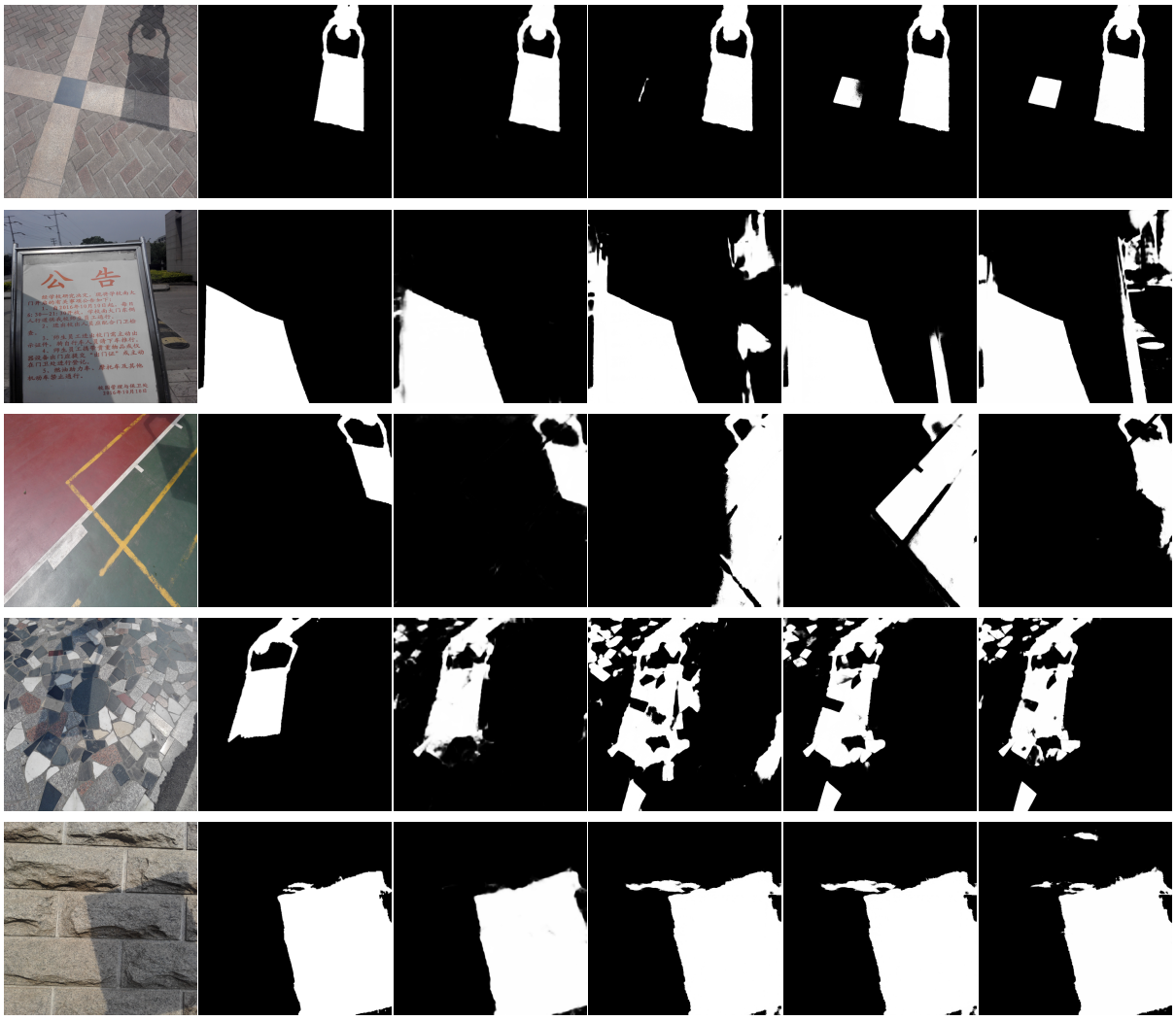
C. Additional Visual Results

We give more visual results of EVP and other task-specific methods on the four tasks in Figure C6, C7, C8, and C9 as supplementary to the visual results in the main paper.



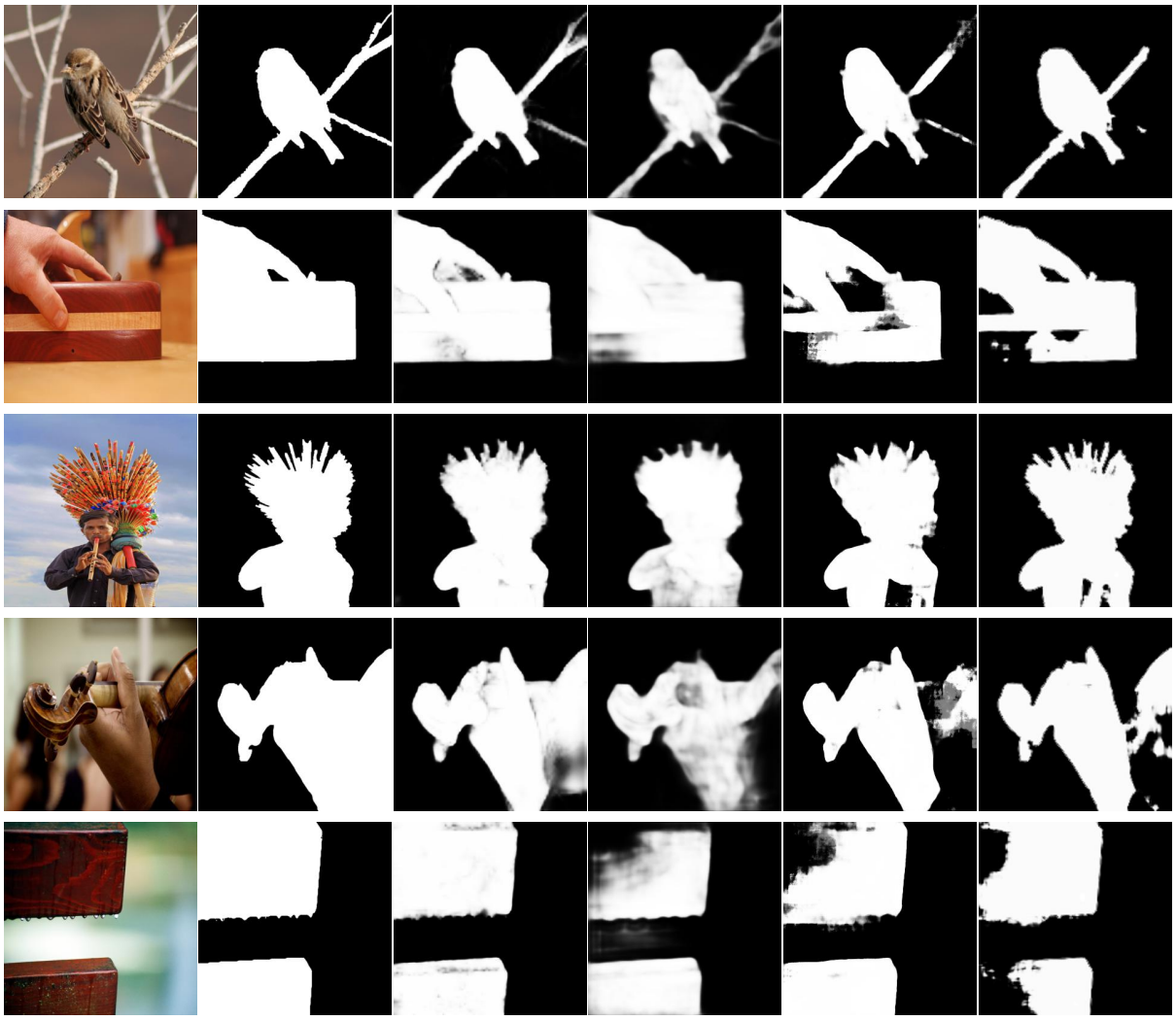
(a) Input (b) GT (c) Ours (d) ManTraNet (e) SPAN

Figure C6. More results on CAISA [12] dataset for forgery detection. We compare to ManTraNet [76] and SPAN [26].



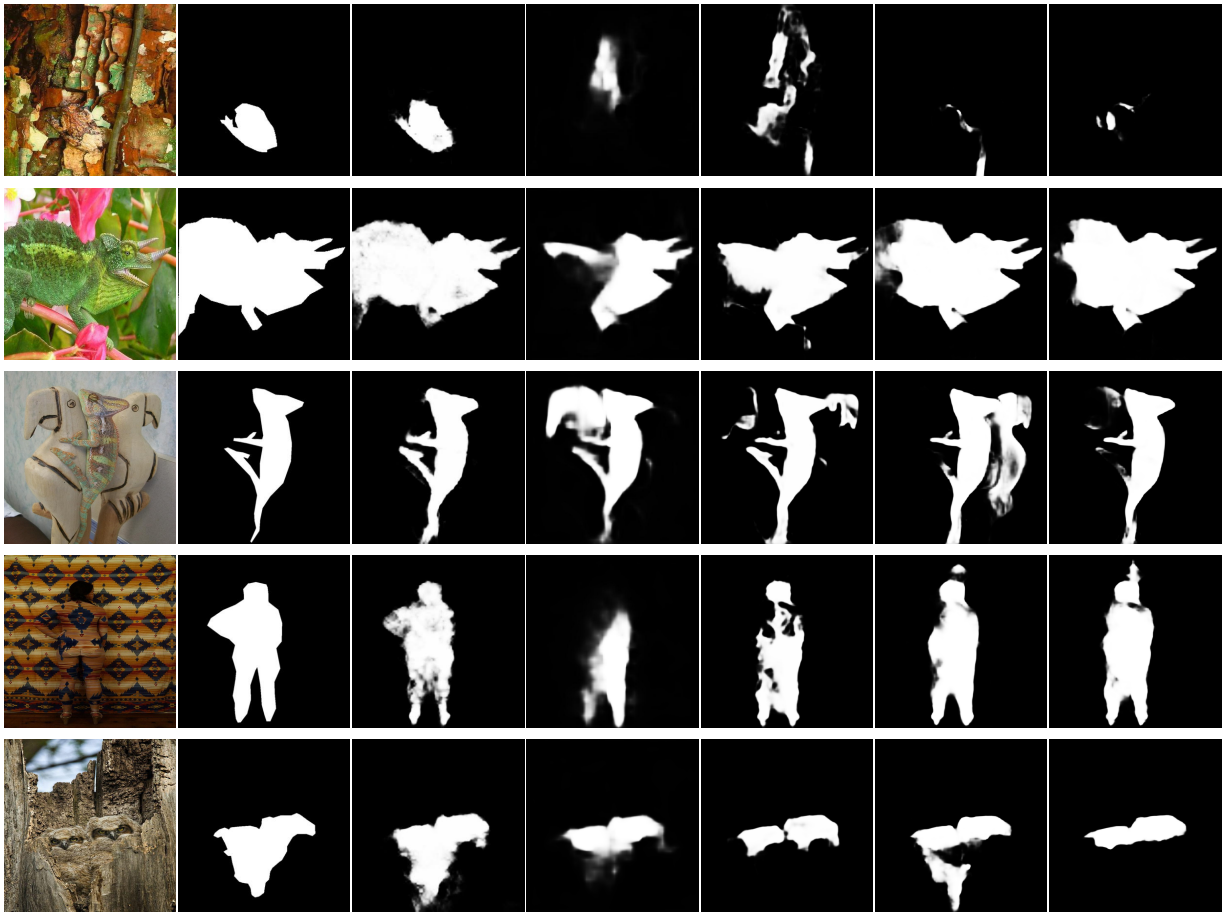
(a) Input (b) GT (c) Ours (d) DSD (e) MTMT (f) FDRNet

Figure C7. More results on ISTD [73] dataset for shadow detection. We compare to DSD [80], MTMT [5], FDRNet [90].



(a) Input (b) GT (c) Ours (d) BTBNet (e) CENet (f) EFENet

Figure C8. More results on CUHK [67] dataset for defocus blur detection.We compare to BTBNet [82], CENet [83] and EFENet [79].



(a) Input (b) GT (c) Ours (d) SINet (e) PFNet (f) JCOD (g) RankNet

Figure C9. More results on CAMO [43] dataset for camouflaged object detection. We compare to SINet [15], PFNEt [54], JCOD [44] and RankNet [51].