GLPI: A Global Layered Prompt Integration approach for Explicit Visual Prompt

Yufei Gao[1,3], Bin Fu[1], Lei Shi[1,3], Chengming Liu[1,3], Yucheng Shi[†2]

[1] School of Cyber Science and Engineering-Zhengzhou University, [2] School of Computer and Artificial Intelligence-Zhengzhou University, [3] SongShan Laboratory-Zhengzhou

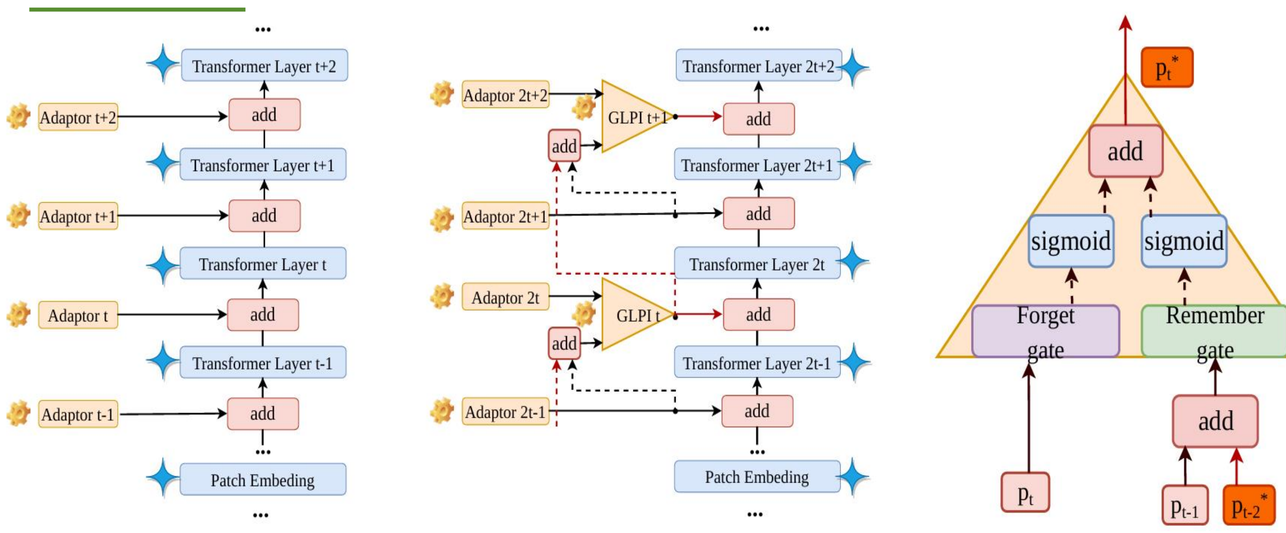
Motivation

- When training large models with EVP, the method of freezing parameters and centrally updating prompt embeddings may bring difficulties to long-distance backpropagation.
- In computer vision, prompt embeddings are not pre-trained and require more optimization steps.
- Inspired by LSTM, long sequences can be processed more effectively and gradients can be propagated more stably by selectively remembering or ignoring information in the input sequence.

Contribution

- Propose Global Layered Prompt Integration (GLPI) for EVP prompt embeddings training. It adjusts embeddings during training by integrating information from different encoder layers, optimizing and enhancing model generalization performance.
- Propose different GLPI variants to balance computational overhead and segmentation performance by adjusting the number and connection of GLPIs.
- GLPI outperforms EVP in six datasets for three tasks and gets better results in some datasets compared to other advanced approaches.

Method



(a) EVP original processing flow, mainly through adaptors to complete the update of the prompts;
(b) The GLPI proposed in this paper provides additional updates to the prompts during the training process by linking the adaptors of different layers;
(c) Detailed internal structure of GLPI. In this paper, yellow screws are used to represent layers that need to be adjusted during training, and blue stars are used to represent layers that are frozen during training.

As shown in the figure on the left, we use p for prompt embedding. p_t represents the original prompt input for the t encoder, p_t^* represents the integration prompt input for the t encoder, p_{t-1} represents the original prompt input for the t-1 encoder, and p_{t-2}^* represents the integration prompt input for the t-2 encoder. Then, p_t, p_{t-1} and p_{t-2}^* are processed to form a complete output for prompt adjustment. Prompts are updated as follows:

$$p_t^* = GLPI(p_t, p_{t-1}, p_{t-2}^*) \quad (1)$$

The following equation demonstrates how GLPI handles prompt embeddings:

$$p_t^* = \sigma_F\left(W_F(p_t)\right) + \sigma_R\left(W_R(p_{t-1} + p_{t-2}^*)\right) \quad (2)$$

The weights of the forget unit and remember unit are represented by W_F and W_R respectively. The activation functions σ_F, σ_R are also included in this equation.

Experiment

- Presented below are the results of our method compared with other advanced methods on six datasets and the experimental results of different variants of our method.

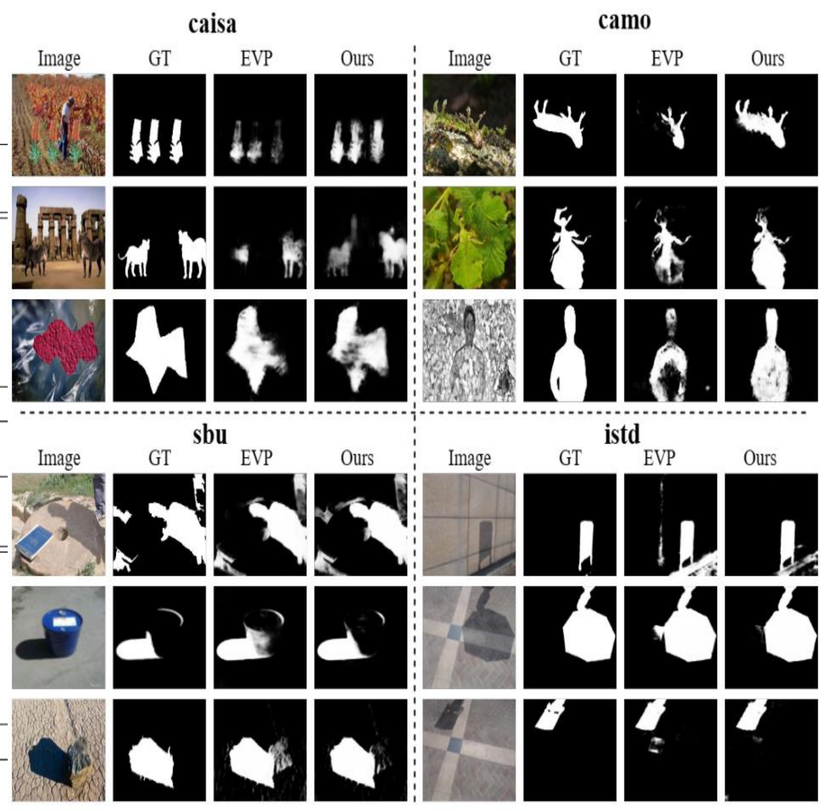
Method	CHAMELEON [26]				CMAO [32]				COD10K [4]			
	$S_\alpha\uparrow$	$E_\phi\uparrow$	$F_\beta^\omega\uparrow$	MAE↓	$S_\alpha\uparrow$	$E_\phi\uparrow$	$F_\beta^\omega\uparrow$	MAE↓	$S_\alpha\uparrow$	$E_\phi\uparrow$	$F_\beta^\omega\uparrow$	MAE↓
RankNet[38]	0.846	0.913	0.767	0.045	0.712	0.791	0.583	0.104	0.767	0.861	0.611	0.045
JCOD[1]	0.870	0.924	-	0.039	0.792	0.839	-	0.082	0.800	0.872	-	0.041
PFNet[6]	0.882	**0.942**	0.810	0.033	0.782	0.852	0.695	0.085	0.800	0.868	0.660	0.040
FBNet[20]	**0.888**	0.939	**0.828**	0.032	0.783	0.839	0.702	0.081	0.809	0.889	0.684	0.035
EVP[34]	0.871	0.917	0.795	0.036	0.846	0.895	0.777	0.059	0.843	0.907	0.742	0.029
Ours	**0.879**	**0.921**	**0.811**	**0.032**	**0.851**	**0.906**	**0.792**	**0.055**	**0.844**	**0.909**	**0.750**	**0.027**

Method	CAISA [8]	
	F1↑	AUC↑
SPAN[37]	0.382	0.838
PSCCNet[22]	0.554	0.875
TransForensics[11]	0.627	0.837
ObjectFormer[12]	0.579	**0.882**
EVP[34]	0.636	0.862
Ours	**0.642**	**0.877**

Method	Trainable Param.(M)	ISTD [10]	CAISA [8]		CAMO [32]			
		BER↓	F1↑	AUC↑	$S_\alpha\uparrow$	$E_\phi\uparrow$	$F_\beta^\omega\uparrow$	MAE↓
Full-tuning	64.00	2.42	0.465	0.754	0.837	0.887	0.778	0.060
Only Decoder	3.15	4.36	0.396	0.722	0.783	0.827	0.671	0.088
VPT-Deep [24]	3.27	1.73	0.588	0.847	0.833	0.884	0.751	0.068
AdaptFormer [29]	3.21	1.85	0.602	0.855	0.830	0.877	0.750	0.068
EVP[34]	3.70	1.35	0.636	0.862	0.846	0.895	0.777	0.059
Ours	4.47	**1.16**	**0.642**	**0.877**	**0.851**	**0.906**	**0.792**	**0.055**

Method	SBU [31]	ISTD [10]
	BER↓	BER↓
DSC[7]	5.59	3.42
DSD[40]	3.45	2.17
MIMT[41]	3.15	1.72
FDRNet[15]	**3.04**	1.55
EVP[34]	4.31	1.35
Ours	**3.86**	**1.16**

Method	Trainable Param.(M)	GLPI Numbers	FLOPs Param.(G)	Integration Prompts	ISTD [10]	CAISA [8]		CMAO [32]			
					BER↓	F1↑	AUC↑	$S_\alpha\uparrow$	$E_\phi\uparrow$	$F_\beta^\omega\uparrow$	MAE↓
EVP[34]	3.70	0	21.69	No	1.35	0.636	0.862	0.846	0.895	0.777	0.059
$Ours_{fc}$	4.47	41	23.42	Yes	1.20	**0.647**	**0.880**	**0.852**	0.906	0.786	0.056
$Ours_{hc}$	4.47	19	22.57	No	1.25	0.636	0.866	0.850	0.901	0.783	0.057
$Ours_{otic}$	4.47	13	22.29	Yes	1.21	0.634	0.870	0.849	0.901	0.784	0.057
Ours	4.47	19	22.57	Yes	**1.16**	0.642	0.877	0.851	**0.906**	**0.792**	**0.055**



- Visualization of EVP and Ours. We show the segmentation results of: EVP and Ours on CAISA dataset for forgery detection (Top-left), on CAMO dataset for camouflaged object detection (Top-right), on SBU and ISTD dataset for shadow detection (Bottom).