Attention to the Burstiness in Visual Prompt Tuning

Yuzhu Wang [1] Manni Duan [1] Shu Kong [2,3]

[1] Zhejiang Lab [2] University of Macau [3] Institute of Collaborative Innovation

ICCV OCT 19-23, 2025 HONOLULU HAWAII

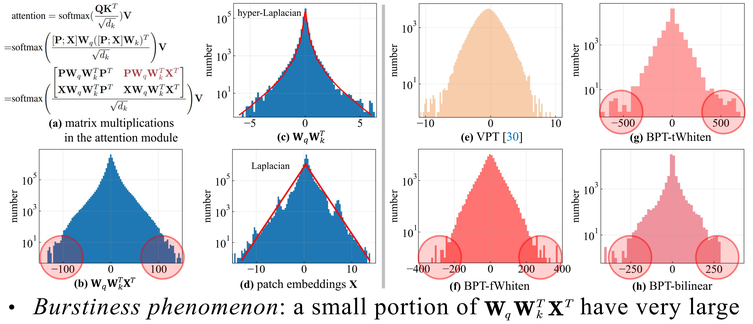
Summary

Visual Prompt Tuning (VPT) is a parameter-efficient fine-tuning technique that learns a small set of parameters in the input space, known as prompts, to adapted a pretrained ViT. In VPT, we uncover a **"burstiness" phenomenon and non-Gaussian distributions** in the values resulting from the interaction of the key and query projectors, and patch embeddings within the self-attention module. Intuitively, these issues pose intuitive challenges for prompt learning.

We address the issues with our proposed **Bilinear Prompt Tuning** (BPT), which either applies ***data whitening*** prior to prompt tuning, or jointly learns ***two compact matrices*** and uses their product as the final prompt.

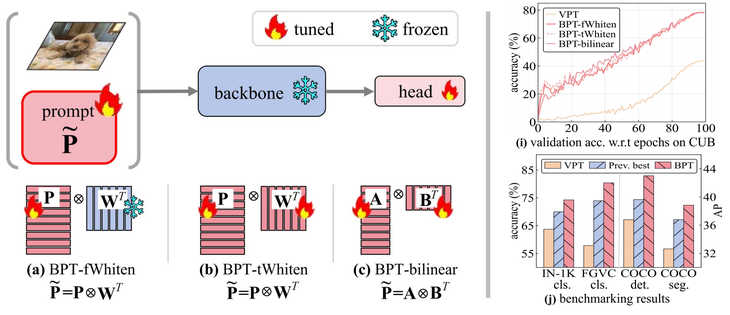
We show that our BPT significantly accelerates learning, reduces parameter count and computation, and importantly achieves the state-of-the-art over various benchmark datasets across model scales, dataset sizes, and pre-training objectives.

Observation and Motivation



- *Burstiness phenomenon*: a small portion of $\mathbf{W}_q\mathbf{W}_k^T\mathbf{X}^T$ have very large absolute values **(b)**.
- *Non-Gaussian distributions*: values of $\mathbf{W}_q\mathbf{W}_k^T$ follow a hyper-Laplacian distribution **(c)**, and values of patch embedding $\mathbf{X}$ follow a Laplacian distribution **(d)**.
- We are motivated to apply whitening to $\mathbf{W}_q\mathbf{W}_k^T\mathbf{X}^T$, transforming it to be more Gaussian before prompt learning. Interestingly, this produces "bursty" prompts **(f)**, but significantly accelerates learning, boosts accuracy **(i)**.

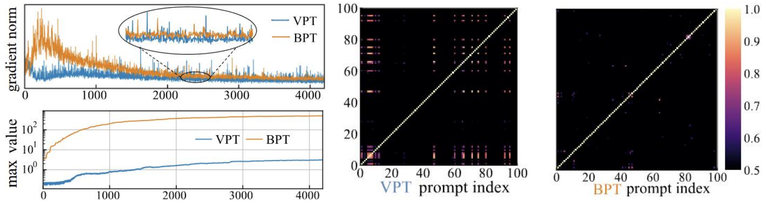
Methods



Overview. To overcome the challenges due to the burstiness and non-Gaussian distributions, we introduce Bilinear Prompt Tuning (BPT) with three approaches, which share the same bilinear form in prompt learning.

- **(a)** and **(b)** apply data whitening $\mathbf{W}$, which transforms the non-Gaussian values of $\mathbf{W}_q\mathbf{W}_k^T\mathbf{X}^T$ to be more Gaussian. Their difference lies in whether tuning $\mathbf{W}$ when learning prompts $\mathbf{P}$.
- **(c)** learns two compact matrices, $\mathbf{A}$ and $\mathbf{B}$, which are multiplied as the final prompts $\mathbf{P}$.

Implementations: We implement the two-matrix multiplication structure in BPT with a 1x1 convolution without bias terms, where the weight is either the whitening matrix $\mathbf{W}$ or the bilinear factor $\mathbf{B}$. The 1x1 layer does **not** adopt any normalizations or nonlinear activations. The other matrix is the prompt $\mathbf{P}$.



Affects on optimization by BPT. We track prompts' gradients norm and max values in optimization iterations. Compared with VPT, whitening helps BPT (left) produce more stable gradients and larger values in learned prompts, (right) BPT produces more de-correlated prompts than VPT.

Experiments

BPT *vs.* VPT *vs.* SPT

(a) Learning "bursty" prompts by our BPT methods outperforms prior arts VPT and SPT. Refer to Fig. 1 for "bursty" distributions.

methods	init.	tuning?	#params	IN-1K	CUB-200
VPT [30]	-	-	7.68	63.71	42.15
SPT [68]	-	-	7.68	69.98	71.15
BPT-fWhiten	whiten	✗	7.68	72.09	77.48
BPT-tWhiten	whiten	✓	66.66	**72.37**	**78.54**
BPT-bilinear	random	✓	**6.51**	72.15	77.86

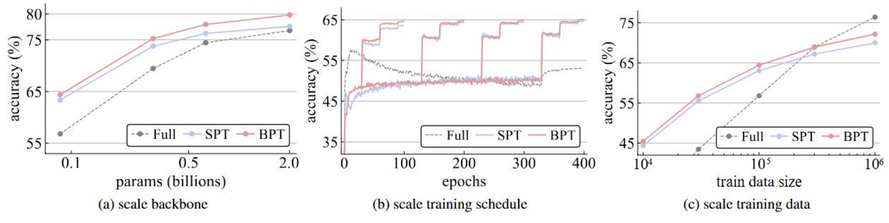
(b) **Prompt length**. Increasing length further improves accuracy.

length	params	IN-1K	CUB-200
64	6.24	71.92	76.91
81	6.37	71.94	77.60
100	6.51	72.15	77.86
144	6.84	72.39	78.36
196	7.23	**72.48**	**78.87**

(c) **Prompt width**. BPT-bilinear learns more compact prompts with higher accuracy.

width	params	IN-1K	CUB-200
SPT [68]	7.68	69.98	71.15
25	2.17	72.06	76.54
50	4.34	72.08	76.95
75	6.51	72.15	77.86
100	8.68	**72.18**	**78.59**

Scale backbone, training iterations, and data



Benchmarking results with different pre-trained models

Methods	Mean Acc	CUB	NABirds	Flowers	Dogs	Cars
MAE pre-training						
Full fine-tuning	82.80	80.55	77.87	91.71	80.38	83.51
VPT-S [30] ECCV'22	57.84	42.15	57.43	69.15	77.07	43.38
SPT-S [68] ICML'24	73.95	71.15	61.87	89.47	80.01	67.23
BPT-S ours	80.39	77.86	72.03	90.37	81.91	79.77
VPT-D [30] ECCV'22	72.02	68.33	65.22	80.05	78.83	67.67
GateVPT [72] ICML'23	73.39	70.56	67.26	78.55	78.90	71.70
SPT-D [68] ICML'24	83.26	80.13	76.28	93.07	82.23	84.61
BPT-D ours	**84.60**	**82.00**	**78.49**	**93.72**	**82.67**	**86.11**
MoCo-V3 pre-training						
Full fine-tuning	84.25	81.75	78.14	94.52	81.19	85.67
VPT-S [30] ECCV'22	79.26	79.05	72.92	90.47	81.97	71.91
SPT-S [68] ICML'24	84.08	83.50	75.79	95.03	84.17	81.93
BPT-S ours	85.05	84.39	76.71	95.84	84.46	83.84
VPT-D [30] ECCV'22	83.12	82.67	75.99	94.41	83.33	79.18
GateVPT [72] ICML'23	83.00	82.86	76.02	93.71	83.37	79.02
SPT-D [68] ICML'24	86.00	84.47	77.63	96.10	85.84	85.98
BPT-D ours	**86.55**	**85.28**	**78.44**	**96.45**	**86.17**	**86.43**

Methods	AP^{box}	AP^{box}_{75}	AP^{box}_s	AP^{mask}	AP^{mask}_{75}	AP^{mask}_s
Mask R-CNN						
Linear probing	30.70	32.44	20.03	28.73	29.83	14.69
VPT-Shallow [29]	33.98	36.45	20.94	31.68	33.21	14.89
SPT-Shallow [61]	36.46	39.46	22.43	33.70	34.21	16.24
BPT-Shallow (ours)	38.55	41.94	23.93	35.35	37.43	17.42
VPT-Deep [29]	34.62	36.82	22.37	32.63	34.41	16.77
SPT-Deep [61]	37.85	41.42	23.51	34.71	36.86	17.22
BPT-Deep (ours)	**39.67**	**43.26**	**24.52**	**36.42**	**38.92**	**17.87**
Cascade Mask R-CNN						
Linear probing	35.12	38.04	21.81	31.29	33.08	16.02
VPT-Shallow [29]	36.78	39.63	21.10	32.63	34.99	14.93
SPT-Shallow [61]	39.63	42.93	24.86	36.78	39.55	19.18
BPT-Shallow (ours)	43.01	46.78	26.94	38.83	41.95	20.70
VPT-Deep [29]	38.70	42.18	22.97	34.27	36.79	16.91
SPT-Deep [61]	41.32	44.80	25.53	37.53	41.54	20.53
BPT-Deep (ours)	**44.97**	**49.42**	**28.10**	**39.69**	**43.17**	**21.36**

$\mathbf{W}_q\mathbf{W}_k^T$ distributions at different layers