Learning Semantic Proxies from Visual Prompts for Parameter-Efficient Fine-Tuning in Deep Metric Learning

ICLR 2024

Li Ren, Chen Chen, Liqiang Wang, Kien Hua

University of Central Florida

Background

- **DML**: map data to embedding space where similar data are closer
- **Proxy-based losses**: learn proxies to represent class
- Pre-trained vision transformers (ViT) have strong representation ability
- Full fine-tuning is computationally expensive, prone to overfitting and catastrophic forgetting

We propose a **parameter-efficient fine-tuning (PEFT)** framework applying the **visual prompt tuning (VPT)** to fine-tune vision transformers for DML.

Visual Prompt Tuning (VPT)

- ❑Learnable prompts append to ViT embeddings
- ❑Only tune prompts + head, rest of ViT frozen
- ❑Highly parameter efficient

Benefits:

- ➢Efficient adaptation
- ➢Retains pre-trained knowledge
- ➢Strong performance in downstream tasks

Proxy-based Deep Metric Learning (DML)

- Learn proxies to represent each class
- Compare samples to proxies instead of each other
- Proxies updated with samples in each batch

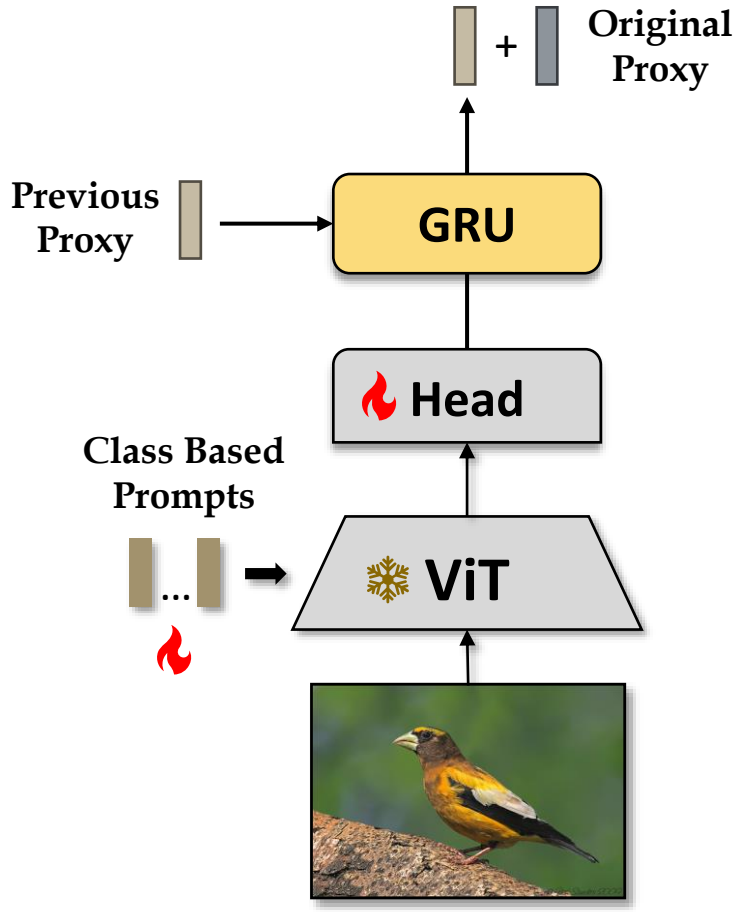
Challenges:

➢Proxies initialized randomly lack semantic info

➢Updating proxies with limited samples in each batch

Our Framework

- ❑Generate semantic proxies using class-specific prompts
- ❑Integrate semantic info into proxies
- ❑Combine with original proxies as bias

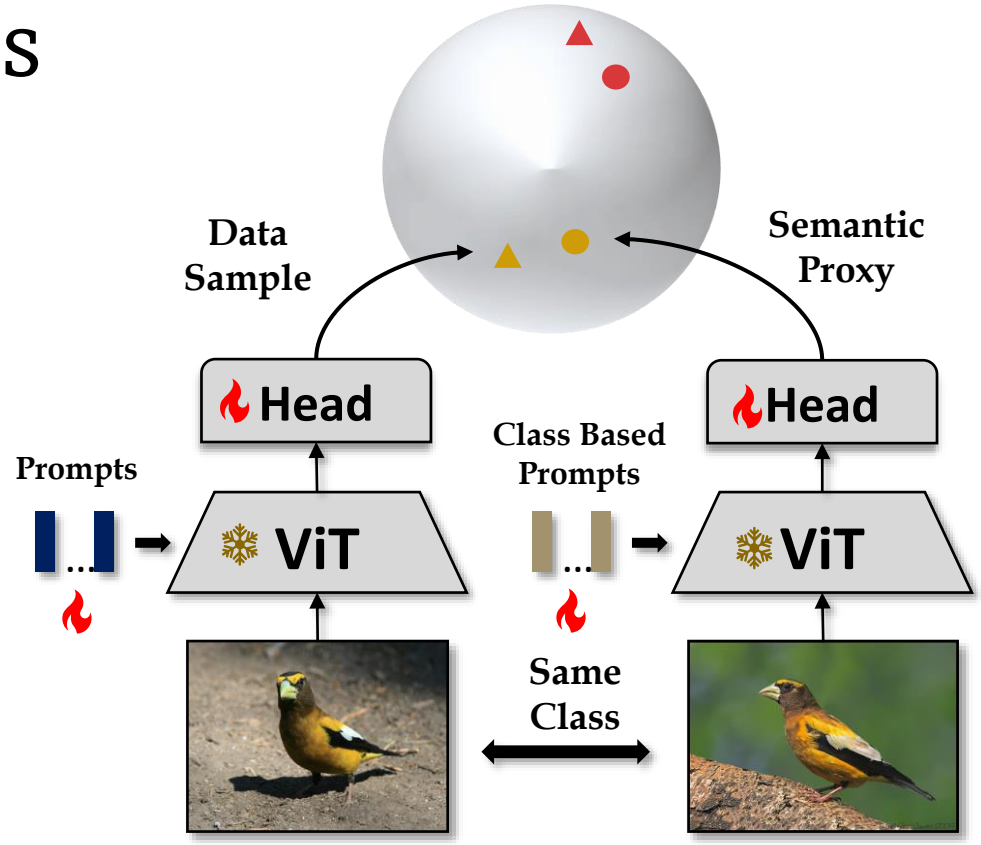


Generating Semantic Proxies

- ❑Generate proxy from ViT with class-specific prompts
- ❑Allows proxies to carry semantic info

Benefits:

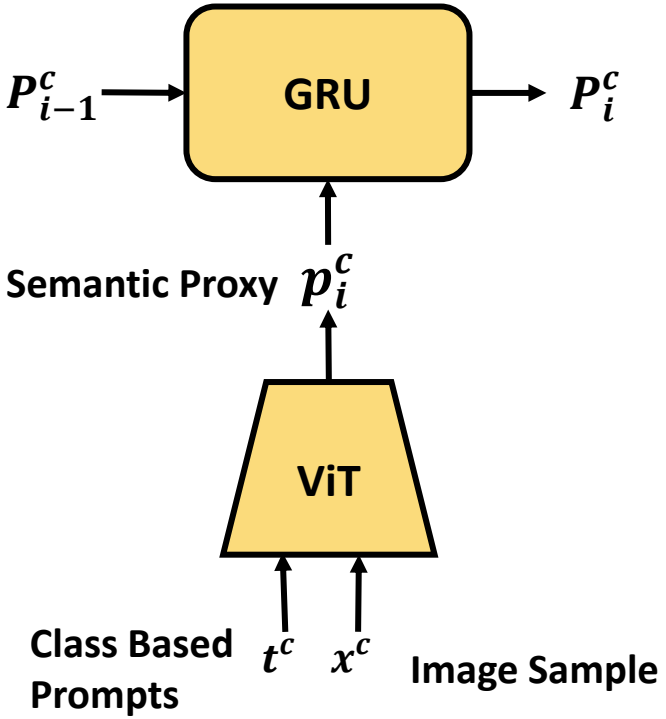
- ➢Proxies have semantic meaning
- ➢Speeds up convergence
- ➢Improves representation quality



Integrating Semantic Proxies

- Use GRU to sequentially integrate proxies
- Allows capturing data structure over time
- Filters out irrelevant noise

$$z_i = \sigma(W_z p_i^c + U_z \mathcal{P}_{i-1}^c + b_z)$$
$$r_i = \sigma(W_r p_i^c + U_r \mathcal{P}_{i-1}^c + b_r)$$
$$\mathcal{N}_i^c = Relu(W_h p_i^c + r_i \odot U_h \mathcal{P}_{i-1}^c + b_h)$$
$$\mathcal{P}_i^c = normal((1 - z_i) \odot \mathcal{P}_{i-1}^c + z_i \odot \mathcal{N}_i^c)$$



Combining with Original Proxies

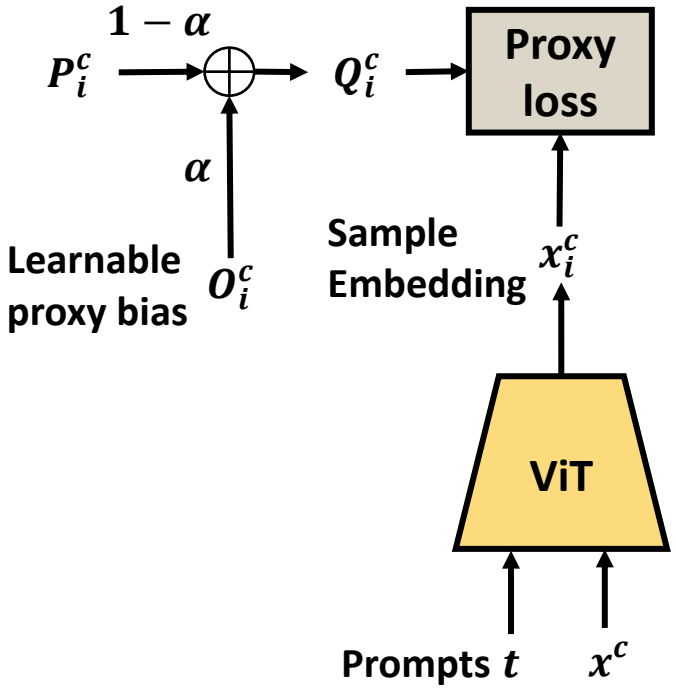
Original proxies as bias term

- ❑Handles unsampled classes
- ❑Balances update frequency vs delay

Final proxy is weighted combination

$$\mathcal{Q}_i^c = normal((1-\alpha)\mathcal{P}_i^c + \alpha\mathcal{O}_i^c)$$

Updated within and across batches



Comparison of PEFT Methods

Method	Compare Architecture and Performance on CUB200			
	Parameter(tunable)	Memory	R1	MAP@R
Full Fine-tuning (PA)	30.2M (100%)	6.0G (100%)	85.5	51.1
Linear Prob	0.19M (0.63%)	1.7G (28.3%)	84.9 (-0.6)	48.1 (-3.0)
BitFit	0.26M (0.86%)	4.4G (73.3%)	85.7 (+0.2)	51.3 (+0.2)
Adapter (L:7,d:256)	1.57M (5.2%)	1.9G (31.7%)	83.1 (-2.4)	46.4 (-4.7)
Adapter (B) (L:1,d:256)	0.46M (1.52%)	2.4G (40%)	85.8 (+0.3)	49.7 (-1.4)
VPT (L:12,N:10)	0.23M (0.76%)	2.2G (36.6%)	85.1 (-0.4)	51.2 (+0.1)
VPT (B) (L:12,N:10)	0.30M (0.99%)	2.2G (36.6%)	85.7 (+0.2)	51.6 (+0.5)
VPTSP-M (ours)	0.60M (1.9%)	2.5G (41.7%)	85.4 (-0.1)	51.4 (+0.3)
VPTSP-M (B) (ours)	0.67M (4.5%)	2.6G (43.3%)	85.9 (+0.4)	51.8 (+0.7)
VPTSP-G (ours)	1.48M (4.9%)	2.5G (41.7%)	86.1 (+0.6)	52.1 (+1.0)
VPTSP-G (B) (ours)	1.56M (5.2%)	2.6G (43.3%)	**86.6 (+1.1)**	**52.7 (+1.6)**

Compare to state-of-the-art

Method	Settings	CUB-200			CARS-196			SOP			InShop		
Name (Batch Size)	Arch/Dim	R@1	R@2	R@4	R@1	R@2	R@4	R@1	R@10	R@100	R@1	R@10	R@20
Hyp-ViT (882/900)	ViT-S16/384	85.6	91.4	94.8	86.5	92.1	95.3	**85.9**	**94.9**	**98.1**	**92.5**	**98.3**	**98.8**
VPT-Base (64/32)	ViT-S16/384	85.1	91.1	94.0	86.3	91.8	95.5	82.1	92.5	97.1	88.4	97.5	98.4
VPTSP-M (64/32)	ViT-S16/384	85.4	91.2	94.6	86.8	92.0	95.5	82.8	93.1	97.3	90.8	97.7	98.6
VPTSP-G (64/32)	ViT-S16/384	**86.6**	**91.7**	**94.8**	**87.7**	**93.3**	**96.1**	84.4	93.6	97.3	91.2	97.6	98.4
RS@K (>200)	ViT-B16/512	–	–	–	89.5	94.2	96.6	**88.0**	**96.1**	**98.6**	–	–	–
VPTSP-G (64/32)	ViT-B16/512	**88.5**	**92.8**	**95.1**	**91.2**	**95.1**	**97.3**	86.8	95.0	98.0	**92.5**	**98.2**	**98.9**

Table 2: Comparison with the state-of-the-art full fine-tuning methods, including Hyp-ViT (Ermolov et al., 2022) and RS@K (Patel et al., 2022) applying ViT on the conventional benchmarks pre-trained on ImageNet-21K. The second column shows the architecture of the backbone and the feature dimension we selected to compare. We also indicate the batch size for training in the brackets. ViT-S16 and ViT-B16 represent the *small* and *basic* size of ViT with patch size 16×16. We shade our proposed method, and the best under the same pretraining and architecture are bold.

Conclusion

- ❑Semantic proxies via class-specific prompt tuning
- ❑GRU integration of proxies over time
- ❑Combining the semantic proxies with original proxies to handle unsampled classes
- ❑Comparable or State-of-the-art results with ~5% of parameters tuned

Prompt tuning is a powerful tool for adapting vision transformers to DML in a parameter-efficient manner

Thanks for watching!

Please download and try our code at:

PAPER DOWNLOAD

TRY OUR CODE