Faster Parameter-Efficient Tuning with Token Redundancy Reduction - Supplementary material -

Kwonyoung Kim[1] Jungin Park[1*] Jin Kim[1] Hyeongjun Kwon[1] Kwanghoon Sohn[1,2*]
[1]Yonsei University, [2]Korea Institute of Science and Technology (KIST)
{kyk12, newrun, kimjin928, kwonjunn01 khsohn}@yonsei.ac.kr

In this supplementary material, we provide additional experiment results including visualizations of merging outcomes and the performance evaluations on few-shot learning tasks, cross-modal retrieval tasks, full datasets, and other backbones to demonstrate the superiority of FPET across diverse settings. We provide pseudo-code implementation and complete source code files to offer a more in-depth understanding of our proposed method. Finally, for clarity, we detail hyper-parameters for our experiments and dataset configurations of the datasets we utilized.

A. Additional Analysis

A.1. Performance on other backbones

We extend the evaluation of our method to include other backbones, such as ViT-S, ViT-L, DeiT-S and DeiT-B, to assess its generalizability across different model architectures. ViT-S and DeiT-S features a token dimension that is half the size of ViT-B. In contrast, ViT-L employs a larger token dimension and consists of 24 transformer layers which is twice deeper layers than ViT-B, signifying a more complex model structure. DeiT-B shares a similar structure with ViT-B but utilizes additional distillation token. In all cases, our method is implemented at the middle layer. Our results, as detailed in Tab. 1, demonstrate that our approach not only maintains very competitive accuracy in comparison to the original implementations but also achieves significant efficiency gains, underscoring the generalizability of our method across different backbone architectures.

Model	Method	Acc (%)	Time (ms)	FLOPs (G)	Mem (GB)
ViT-S	Bi-LoRA [16]	74.9	0.67	4.7	5.0
	FPET-Bi-LoRA	74.9	0.59 (-15.8%)	3.6 (-23.6%)	4.3 (-13.8%)
ViT-L	LoRA [13]	76.0	8.67	61.8	19.6
	FPET-LoRA	76.0	6.60 (-23.9%)	46.7 (-24.4%)	15.5 (-21.0%)
DeiT-S	LoRA[13]	70.1	1.08	4.8	5.0
	FPET-LoRA	70.0	0.93(-14.2%)	3.6(-23.5%)	4.3(-13.9%)
	Bi-LoRA [16]	70.2	1.08	4.8	5.0
	FPET-Bi-LoRA	70.2	0.93 (-14.2%)	3.6 (-23.5%)	4.3 (-13.9%)
DeiT-B	LoRA [13]	72.9	2.62	17.6	8.4
	FPET-LoRA	72.8	2.10 (-18.7%)	13.3 (-24.4%)	7.1 (-15.5%)
	AdaptFormer[6]	72.7	2.73	17.7	7.7
	FPET-AdaptFormer	72.6	2.15(-21.9%)	13.5(-21.9%)	6.2(-21.9%)

Table 1. Model performance on other backbones.

A.2. Comparison with rank redundancy reduction

We compare rank redundancy reduction with token reduction by benchmarking our method against a model with a hidden dimension of 1. As shown in Tab. 2, token reduction demonstrates superior performance in both accuracy and efficiency. Despite a significant reduction in the hidden dimension, the gains in efficiency are limited, while there is a notable decrease in performance. In contrast, token reduction not only offers greater efficiency improvements but also maintains accuracy. Therefore, in the context of enhancing efficiency, reducing token redundancy emerges as a significantly more effective strategy than reducing rank.

Method	Acc (%)	Time (ms)	FLOPs (G)	Mem (GB)
AdaptFormer (dim=8)	76.2	2.68	17.61	7.64
AdaptFormer (dim=1)	74.7	2.64	17.59	7.62
FPET-AdaptFormer	76.2	2.12	13.45	6.21

Table 2. Comparison between reducing rank redundancy and token redundancy.

A.3. Trade-offs between efficiency and accuracy

We provide numerical data corresponding to Fig 3. of the main paper in Tab. 3. As shown in Tab. 3, our original implementation, which operates at layer 6, achieves a 20.9% reduction in inference time without compromising accuracy compared to the original AdaptFormer [6] implementation. Furthermore, greater efficiency gains can be achieved by applying our module to earlier layers.

We present the same experimental results on Bi-AdaptFormer [16] in Tab. 4 and Fig. 1. Compared to the original Bi-AdaptFormer, our implementation at layer 6 achieves a 23.3% reduction in FLOPs. By applying our module at layer 4, we achieve a 31.8% reduction in FLOPs while maintaining competitive accuracy relative to state-of-the-art PET and efficiency-focused PET methods, as shown

* Corresponding authors.

Layer	Method	Acc (%)	Time (ms)	FLOPs (G)
N/A	w/o merging	76.22	2.68	17.6
6	Max Pool.	75.68	2.06	13.5
	Avg. Pool.	75.87	2.08	13.5
	Bipartite soft matching [3]	76.03	2.12	13.5
	Bipartite differentiable matching	76.22	2.12	13.5
4	Max Pool.	75.03	1.89	12.0
	Avg. Pool.	75.13	1.89	12.0
	Bipartite soft matching [3]	75.40	1.92	12.0
	Bipartite differentiable matching	75.65	1.92	12.0
2	Max Pool.	73.41	1.67	10.5
	Avg. Pool.	73.43	1.68	10.5
	Bipartite soft matching [3]	73.95	1.71	10.5
	Bipartite differentiable matching	74.29	1.71	10.5
1	Max Pool.	71.46	1.56	9.8
	Avg. Pool.	71.45	1.57	9.8
	Bipartite soft matching [3]	71.06	1.58	9.8
	Bipartite differentiable matching	72.21	1.59	9.8
0	Max Pool.	68.95	1.40	9.0
	Avg. Pool.	69.33	1.42	9.0
	Bipartite soft matching [3]	67.39	1.44	9.0
	Bipartite differentiable matching	69.09	1.45	9.0

Table 3. Trade-off between inference time and accuracy. All models are ViT-B/16 consisting of 12 transformer layers with AdaptFormer [6]. All methods reduce 196 tokens to 98 tokens at different layers.

Layer	Method	Acc (%)	Time (ms)	FLOPs (G)
N/A	w/o merging	77.01	2.77	17.7
6	Max Pool.	76.65	2.13	13.5
	Avg. Pool.	76.79	2.14	13.5
	Bipartite soft matching [3]	76.83	2.17	13.5
	Bipartite differentiable matching	76.96	2.17	13.5
4	Max Pool.	76.06	1.92	12.0
	Avg. Pool.	76.22	1.91	12.0
	Bipartite soft matching [3]	76.50	1.94	12.0
	Bipartite differentiable matching	76.34	1.94	12.0
2	Max Pool.	74.26	1.71	10.5
	Avg. Pool.	74.66	1.71	10.5
	Bipartite soft matching [3]	74.96	1.72	10.5
	Bipartite differentiable matching	75.18	1.71	10.5
1	Max Pool.	72.23	1.58	9.8
	Avg. Pool.	72.41	1.59	9.8
	Bipartite soft matching [3]	72.02	1.60	9.8
	Bipartite differentiable matching	73.00	1.60	9.8
0	Max Pool.	69.68	1.51	9.0
	Avg. Pool.	70.31	1.52	9.0
	Bipartite soft matching [3]	67.98	1.54	9.0
	Bipartite differentiable matching	69.75	1.53	9.0

Table 4. Trade-off between inference time and accuracy. All models are ViT-B/16 consisting of 12 transformer layers with BiAdaptFormer [16]. All methods reduce 196 tokens to 98 tokens at different layers.

in Tab. 1 of the main paper. At layer 2, we further reduce FLOPs by 40.6%, demonstrating the scalability and efficiency of our approach.

Compared to the bipartite soft matching [3], our method consistently demonstrates higher accuracy, highlighting the more optimal matching achieved by our approach. This is particularly evident in early layers, where the refinement of similarity is crucial. In these layers, the accuracy of bipartite soft matching [3] drops significantly, performing even worse than basic pooling methods, further underscoring the effectiveness of our method.

A.4. Performance on full datasets

In line with previous studies [14–16, 22, 33], our models are trained on VTAB-1K [32] training set which comprises subset of each original downstream dataset. We extend the evaluation of our method to training on full datasets of CIFAR100 [19] and SVHN [25], which contain 50,000 and 73,257 training images, respectively.

As outlined in the main paper, our implementation utilizes the ViT-B/16 as the backbone model, with the hidden dimension set to 8 for both AdaptFormer [6] and LoRA [13]. As shown in Tab. 5, our method achieves competitive accuracy relative to the original implementations underscoring the robustness of FPET in full dataset scenarios.

A.5. Visualization

In Fig. 3, we visualize token merging results produced by ToMe [3]and our token redundancy reduction module to

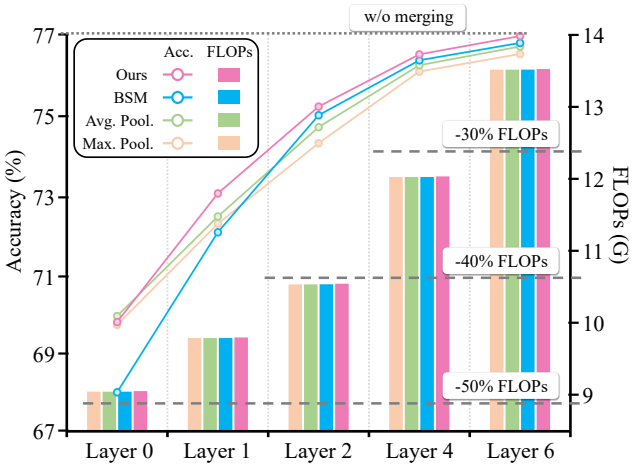


Figure 1. Trade-off between flops and accuracy. All models are ViT-B/16 consisting of 12 transformer layers with BiAdaptFormer [16]. BSM refers to bipartite soft matching [3]. All methods reduce 196 tokens to 98 tokens at different layers.

Method	CIFAR-100	SVHN
AdaptFormer	92.14	97.21
FPET-AdaptFormer	92.18	97.18
LoRA	92.17	97.45
FPET-LoRA	92.29	97.48

Table 5. Accuracy on full CIFAR-100 and SVHN dataset.

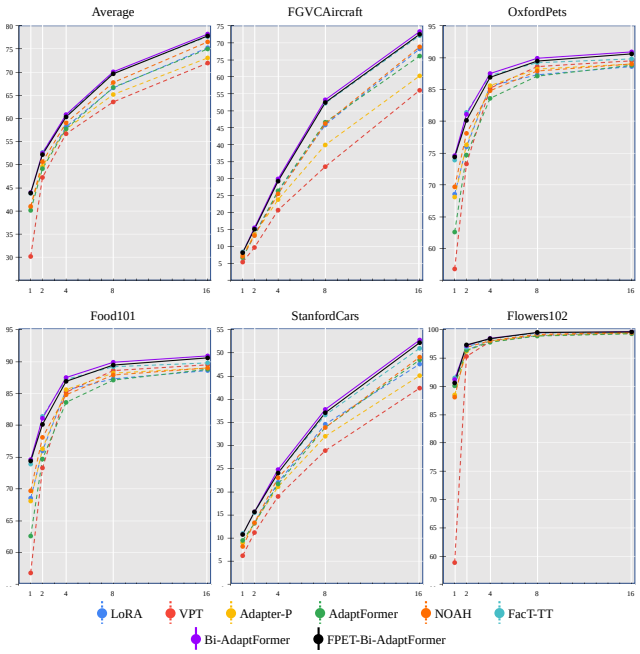


Figure 2. Performance of Few-shot learning on FGVC dataset including FGVC-Aircraft [24], Oxford-Pets [27], Food-101 [5], Stanford Cars [18] and Oxford-Flowers102 [26].

demonstrate the effectiveness of the proposed method. Employing the visualization techniques in [3], we trace the trajectory of each token and project the merging results onto the original image by averaging the colors within each group. This method ensures that patches belonging to the same merging group are represented by the same averaged color and bordered by a color randomly assigned to each group, facilitating a clear visual distinction.

Notably, whereas ToMe merges 8 tokens across all 12 layers, resulting in a total of 96 tokens to be merged, FPET executes a one-time merge of 98 tokens, equivalent to half of the total patch count, at the middle layer only. Despite the larger number of remaining tokens in the last layer, the visual evidence in Fig. 3 reveals that ToMe[3] often produces images with blurred boundaries and distortions as highlighted in red circles. These results indicate semantically irrelevant merges between the object patches and background or other object patches. In contrast, FPET maintains sharper edges and more authentic shapes, demonstrating more optimal merging outcomes to preserve crucial visual information. This comparison underscores our module delicately engineered for PET and the enhanced precision and effectiveness of the merging process in FPET.

A.6. Accuracy Drop of Other Token Reduction Methods

We report the accuracy drop relative to the original implementations of various token reduction methods across different PET methods. As shown in Tab. 6, our method consistently achieves smaller accuracy drops compared to existing approaches.

Method	LTMP [4]	ToMe [3]	Ours
RepAdapter	-0.32	-0.33	**-0.05**
LoRA [13]	-0.32	-0.32	**-0.12**
AdaptFormer [6]	-0.58	-0.34	**0.00**
Bi-LoRA [16]	-0.34	-0.34	**-0.17**
Bi-AdaptFormer [16]	-0.38	-0.27	**-0.05**

Table 6. Accuracy drop on other token reduction methods. All models are ViT-B/16 consist of 12 transformer layers with respective PET method. Each model starts with 197 tokens. LTMP reduces a variable number of tokens, while ToMe merges 8 tokens at each of the 12 layers.

B. Performance on Other Tasks

B.1. Performance on Few-shot learning

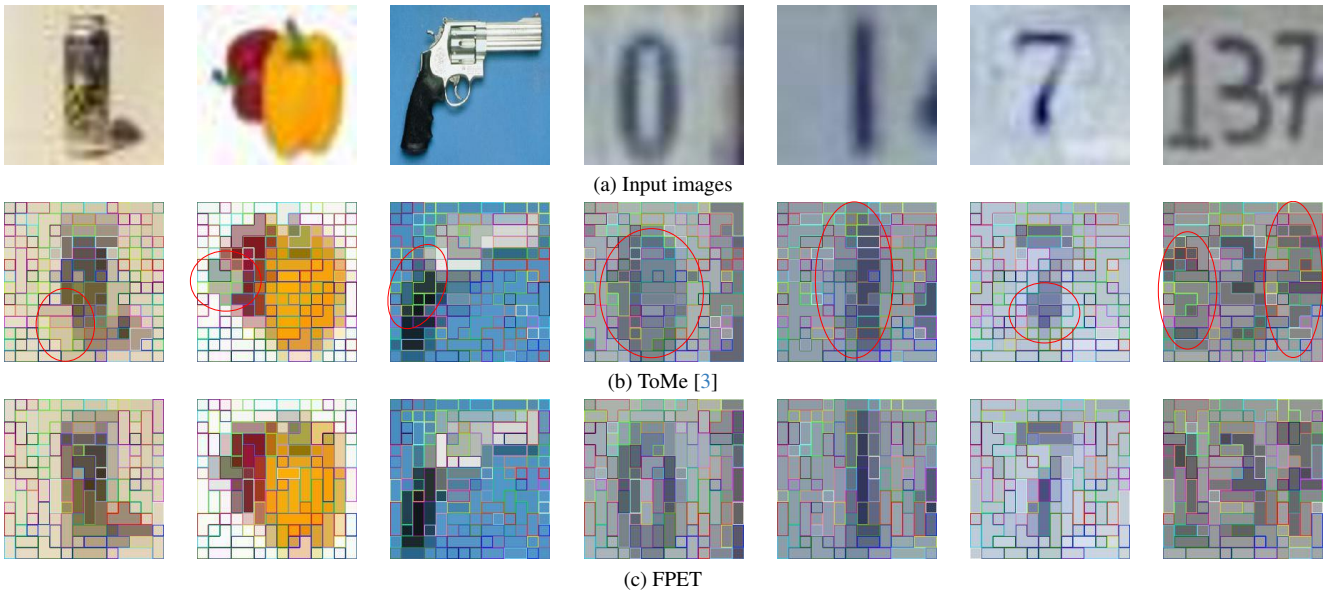
To assess performance of FPET in low-data scenarios, we conduct experiments using the FGVC dataset. Following the setting in [16], we utilize five datasets, FGVC-Aircraft [24], Oxford-Pets [27], Food-101 [5], Stanford Cars [18], and Oxford-Flowers102 [26]. Our experiments span 1, 2, 4, 8, and 16-shot settings.

We implement FPET on Bi-AdaptFormer[16] and compare our Bi-AdaptFormer-FPET with several state-of-the-art approaches, including LoRA[13], VPT[14], Adapter-P[28], AdaptFormer[6], NOAH[33], FacT-TT[15], and Bi-AdaptFormer[16]. As in [16], we configure the hidden dimension as 8 for AdaptFormer [6], LoRA [13], and Adapter-P[28], and as 32 for Bi-AdaptFormer [16] and Bi-AdaptFormer-FPET. The prompt length for VPT is set to 8, while the rank for FacT-TT[15] is determined to be 16. For NOAH[33], we adopt the best configuration as suggested in their paper.

As depicted in Fig. 2, Bi-AdaptFormer [16] emerges as the top performer regarding accuracy. Our approach, Bi-AdaptFormer-FPET, demonstrates competitive accuracy in comparison, thereby underscoring the robustness of FPET in scenarios characterized by limited data availability.

B.2. Performance on Cross-modal Retrieval

We present the cross-modal retrieval performance evaluated on the Flickr30K dataset [29], to further demonstrate the scalability of our method on vision-language tasks. Specifically, we employ the pretrained BLIP-base [21] as our vision-language backbone and apply the proposed FPET to the vision model. As most of the latency arises from the vision model, applying our method to only the pre-trained vision model sufficiently yields significant efficiency im-



(a) Input images

(b) ToMe [3]

(c) FPET

Figure 3. Token merging visualization. We visualize input patches associated with each merged token at the end of networks, following the methodology provided by [3]. Patches belonging to the same merging group are represented by the same averaged color and bordered by a color randomly assigned to each group. Blurred boundaries and distortions indicating semantically irrelevant merging are highlighted in red circles. We employ ViT-B/16 as the backbone and Bi-AdaptFormer as the PET method.

provements. As a baseline, we compare the results with UniAdapter [23] that shares the same spirit of our method. As shown in Tab. 7, we achieve a 28.44% reduction in training time and a 22.62% reduction in GPU memory usage, demonstrating the practicality of our approach in resource-constrained environments.

C. Pseudo code

To succinctly present the implementation of our token redundancy reduction module, we provide its pseudo code representation in Fig. 4.

D. Hyper-parameters

We provide the hyper-parameters for our experiments in Appendix D.

E. Datasets

We provide the statistics of datasets used for our experiments in Tab. 9.

Method	I2T Retrieval			T2I Retrieval			Efficiency	
	R@1	R@5	R@10	R@1	R@5	R@10	Training time (hr/epoch)	Mem (GB)
UniAdapter [23]	94.2	99.5	99.7	83.6	96.6	98.2	1.09	44.96
FPET-UniAdapter	94.1	99.4	99.9	83.0	96.0	98.0	0.78(-28.44%)	34.79(-22.62%)

Table 7. Performance on cross-modal retrieval evaluated on Flickr30K [29]. We report R@1, 5, and 10 for image-to-text (I2T) and text-to-image (T2I) retrieval, and efficiency in terms of training time and memory.

```python
import torch

# bipartite differentiable matching
def bdm(self, tokens, key):

    # halt propagation of gradients to the backbone
    key = key.detach()

    # key refinement
    key_refined = key + self.refinement(key)

    # split tokens
    k_a, k_b = checkerboard_split(key_refined)
    x_a, x_b = checkerboard_split(tokens)

    # refined similarity matrix
    scores = k_a @ k_b.transpose(-1, -2)

    # exclude cls token
    scores[..., :, 0] = -math.inf

    # average of the top-1 and top-2 values
    v, idx = torch.topk(scores, 2, dim=-1)
    mean12 = v.mean(dim=-1, keepdim=True)

    # generate the soft matching matrix
    soft_matrix = torch.sigmoid(scores - mean12)

    # generate the hard matching matrix
    hard_matrix = (soft_mask > 0.5).float()

    # generate the matching matrix
    matching_matrix = soft_matrix + (hard_matrix - soft_matrix).detach()

    # merging tokens
    x_merged_sum = x_b + torch.einsum('bik, bij->bkj', matching_matrix, x_a)
    self.size = self.size_update(self.size, matching_matrix)
    x_merged = self.average(x_merged_sum, self.size)


    return x_merged
```

Figure 4. Implementation our proposed bipartite differentiable matching.

optimizer	batch size	learning rate	weight decay	# epochs	lr decay	# warm-up epochs
AdamW	64	1e-3	1e-4	100	cosine	10

Table 8. Hyper-parameters for our experiments.

	Dataset	# Classes	Train	Val	Test
		VTAB-1K [32]			
Natural	CIFAR100 [19]	100	800/1,000	200	10,000
	Caltech101 [9]	102			6,084
	DTD [8]	47			1,880
	Oxford-Flowers102 [26]	102			6,149
	Oxford-Pets [27]	37			3,669
	SVHN [25]	10			26,032
	Sun397 [31]	397			21,750
Specialized	Patch Camelyon [30]	2	800/1,000	200	32,768
	EuroSAT [11]	10			5,400
	Resisc45 [7]	45			6,300
	Retinopathy [1]	5			42,670
Structured	Clever/count [17]	8	800/1,000	200	15,000
	Clever/distance [17]	6			15,000
	DMLab [2]	6			22,735
	KITTI-Dist [10]	4			711
	dSperites/location [12]	16			73,728
	dSperites/orientation [12]	16			73,728
	SmallNORB/azimuth [20]	18			12,150
	SmallNORB/elevation [20]	18			12,150
		Few-shot learning			
	Food-101 [5]	101	1/2/4/8/16 per class	20,200	30,300
	Stanford Cars[18]	196		1,635	8,041
	Oxford-Flowers102[26]	102		1,633	2,463
	FGVC-Aircraft[24]	100		3,333	3,333
	Oxford-Pets[27]	37		736	3,699
		Full datasets			
	CIFAR100[19]	100	60,000	-	10,000
	SVHN[25]	10	73,257	-	26,032

Table 9. Statistics of datasets.

References

[1] Kaggle and eyepacs. kaggle diabetic retinopathy detection. 2015. 6

[2] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016. 6

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023. 2, 3, 4

[4] Maxim Bonnaerens and Joni Dambre. Learned thresholds token merging and pruning for vision transformers. In *TMLR*, 2023. 3

[5] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, pages 446–461, 2014. 3, 6

[6] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, pages 16664–16678, 2022. 1, 2, 3

[7] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, pages 1865–1883, 2017. 6

[8] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014. 6

[9] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPRW*, 2004. 6

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, pages 1231–1237, 2013. 6

[11] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pages 2217–2226, 2019. 6

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017. 6

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 1, 2, 3

[14] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022. 2, 3

[15] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *AAAI*, pages 1060–1068, 2023. 3

[16] Shibo Jie, Haoqing Wang, and Zhi-Hong Deng. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *ICCV*, pages 17217–17226, 2023. 1, 2, 3

[17] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, pages 2901–2910, 2017. 6

[18] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *CVPRW*, 2013. 3, 6

[19] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 2, 6

[20] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, pages II–104, 2004. 6

[21] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 3

[22] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In *NeurIPS*, pages 109–123, 2022. 2

[23] Haoyu Lu, Yuqi Huo, Guoxing Yang, Zhiwu Lu, Wei Zhan, Masayoshi Tomizuka, and Mingyu Ding. Uniadapter: Unified parameter-efficient transfer learning for cross-modal modeling. In *ICLR*, 2024. 4, 5

[24] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 3, 6

[25] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPSW*, 2011. 2, 6

[26] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *CVPR*, pages 1447–1454, 2006. 3, 6

[27] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012. 3, 6

[28] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *ACL*, pages 487–503, 2021. 3

[29] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, pages 2641–2649, 2015. 3, 5

[30] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *Medical Image Computing and Computer Assisted Intervention*, pages 210–218, 2018. 6

[31] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492, 2010. 6

[32] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. The visual task adaptation benchmark. In *arXiv preprint, arXiv:1910.04867*, 2019. 2, 6

[33] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. In *arXiv preprint, arXiv:2206.04673*, 2022. 2, 3