Parasite Networks: Transfer Learning in Resource-Constrained Domains

Andrew Alini, Douglas Sturim, Kevin Brady, Pooya Khorrami
MIT Lincoln Laboratory
Lexington, MA
{andrew.alini,sturim,kbrady,pooya.khorrami}@ll.mit.edu

Abstract

The effective and efficient transfer of knowledge from a foundation model using Convolutional Neural Networks (CNNs) to a new task is a significant challenge in computer vision. Initial approaches include side-tuning, a straightforward method that attaches a lightweight, modular side-network to the original model and interpolates the two outputs for transference to the new task. They fell to the wayside when visual prompt tuning (VPT) was introduced due to VPT's superior performance, significantly fewer additional parameters, and improved ability to generalize to multiple datasets. This paper presents the Parasite Network, an alternative side-network approach using CNNs that leverages a small 'parasite' model that extracts knowledge at points along the original larger 'host' network without adapting the original model. We show that parasite networks have a significant reduction in the number of training parameters and are able to generalize across multiple datasets as compared to side-tuning. The parasite approach was experimentally validated against both substitutive and additive transfer learning methods using various VTAB-1K datasets. We show that the parasite approach outperforms VPT for CNNs and has superior GPU utilization and competitive latency.

1 Introduction

Deep learning has seen explosive growth over recent years utilizing foundation models that have achieved inspiring results across a broad range of tasks. In computer vision, numerous large models with broad-purpose understandings of images in various domains were developed using Residual Networks [14] and Vision Transformers [12]. These large models, trained on millions or billions of images using extensive computational resources, are useful backbones for transfer learning.

With the parameter space of state of the art (SOTA) systems continuing to grow exponentially, challenges emerge with constrained resource requirements such as storage, latency and GPU utilization. Therefore, developing techniques that can modularly leverage foundation models for multiple tasks becomes critical for resource-constrained environments. Popular main transfer learning techniques can be broken down into two main types: substitutive and additive methods [30]. Substitutive methods

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Department of Defense under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense.

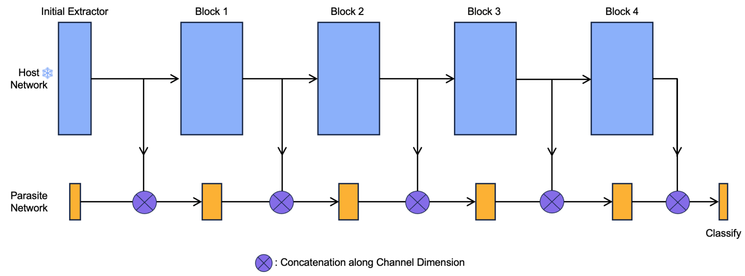

Figure 1: Host Network (ResNet-50) & Parasite Network (ResNet-50 Inspired) Interaction Architecture: The host network is frozen during training with the parasite network having the only trainable parameters. At various predefined points, the output from the host network is concatenated as additional channels to the parasite network which the parasite uses for domain transfer.

are techniques that modify only the parameters of the pretrained network for adaptation to the new task. While this can be an effective approach, the model's knowledge about the original task is lost, which negatively impacts performance in multitask settings. Additive approaches involve freezing the original model and adding additional parameters for domain transfer. This has the advantage of adding fewer additional parameters for each new task enabling efficient multitask learning.

Given these challenges, there is a need for efficient transfer learning approaches in resource-constrained environments. Therefore, we propose parasite networks, a novel side-network approach for transfer learning that is efficient in terms of parameters, GPU memory usage, and latency. The parasite approach can maintain these efficiencies as well as obtain strong performance across multiple VTAB-1K datasets. In this paper, we explore how parasite networks can extract information from foundation (host) models and their advantages over previous transfer learning methods. We conclude that the parasite technique is a strong alternative to the transfer learning problem for CNNs.

2 Related Work

Large Foundation Models for CNNs. Many state-of-the art computer vision foundation models for CNNs rely on Residual Network (ResNet) architectures. ResNets contain skip connections between convolutional layers and many papers use ResNets as the core backbone architecture for many CNN-based foundation models [14, 26, 15, 27]. To train these models, many papers focus on self-supervised learning (SSL) approaches to achieve SOTA results. SSL is mainly broken down into two types: 1) contrastive learning which learns from positive and negative examples [24, 8] and 2) non-contrastive learning which uses only positive examples [5, 7]. While these foundation models provide impressive results, there is a need for transfer learning techniques on these models in resource-constrained environments.

Transfer Learning. In transfer learning, a pretrained model is taken from one task and adapted to another domain. Popular transfer learning techniques can be broken down into two main approaches: substitutive methods and additive methods. Three popular additive techniques are 1) the linear probe approach (Linear Probe-k), 2) side-tuning, and 3) visual prompt tuning (VPT). Linear Probe-k involves using all layers except for the classification layer(s) of a pretrained model as a backbone and then attaching k layers at the end of the network for transfer to the new task [20, 28]. In the side-tuning approach, a small "side" model is trained by combining the core original model's and the smaller model's output representations and classifying from the resultant combination [30]. Finally in VPT, the core model is frozen and additional prompt parameters are added in the input space for adaptation to the new domain [17].

In substitutive approaches, only the original network's parameters are used for transferring to the new domain. Three popular substitutive techniques are: 1) fully retraining the model, 2) partially fine-tuning the model (Partial-k), and 3) bias-tuning. Under the full-retrain technique, the entire model is adapted to the new task [1, 11]. In Partial-K, the last k layers are tuned while the remaining layers are frozen [2, 25]. In the last technique, only the bias vectors are adapted [3, 4].

		Avg. # Trainable Parameters	Datasets						Averages	
			Pet	Flowers	EuroSAT	CIFAR100	Dmlab	DTD	Accuracy	STD
Substitutive	Full-Retrain	23.6M	83.09%	74.26%	90.56%	32.88%	41.41%	50.53%	62.12%	0.84
	Partial-1	4.6M	86.34%	83.44%	90.52%	44.46%	39.02%	62.91%	67.78%	0.52
	Tinytl-bias	129.7K	**88.28%**	80.38%	92.12%	48.38%	31.42%	60.62%	66.87%	0.41
Additive	Linear Probe-1	4.3M	87.50%	80.88%	91.02%	48.13%	34.96%	58.46%	66.83%	0.44
	Linear Probe-2	8.5M	86.41%	80.52%	90.26%	47.42%	35.79%	59.17%	66.6%	0.50
	Linear Probe-4	16.9M	84.18%	69.47%	89.66%	38.66%	32.03%	58.05%	62.01%	1.02
	Linear Probe-8	33.7M	79.49%	58.92%	82.41%	31.59%	26.23%	52.53%	55.2%	1.23
	VPT	117.9K	88.26%	80.93%	90.36%	49.48%	36.29%	63.40%	68.12%	0.17
	Side-tune	21.4M	85.41%	23.03%	82.42%	43.84%	24.88%	60.23%	53.3%	7.80
	Parasite 2.61%	617K	87.20%	83.28%	95.09%	**49.76%**	44.34%	**64.08%**	70.63%	0.61
	Parasite 7.44%	1.8M	87.53%	**85.04%**	**95.41%**	49.51%	**45.99%**	63.81%	**71.22%**	0.49
	Zero-ed Host Parasite	617K	6.72%	18.64%	80.17%	8.96%	35.07%	14.13%	27.28%	0.60

Table 1: Parasite networks with sizes 2.61% and 7.44% of a ResNet-50 were chosen for reporting results. Zero-ed Host Parasite refers to a parasite model that receives no information from the host and must adapt alone. We provide the average number of trainable parameters, accuracy, standard deviation over the six datasets to show the relative differences between runs. We took an average for the number of parameters as the classification heads changes with number of classes.

3 Model Architecture

The overall model architecture is comprised of two parts: 1) a larger host model from the original network and 2) a smaller parasite model. Figure 1 shows how the parasite extracts information after various blocks of the host network. For a more comprehensive architecture design and use case, refer to the supplementary materials.

Host Network The host network corresponds to the large foundation model that contains information for domain transfer. During training and testing, the host network remains frozen and acts as input for the parasite network. Activation outputs are stored at various points along the network with connections at these points to feed information into the parasite model. Having a host model separate from the parasite allows for systems to still utilize the host model for separate purposes as well as for the specific domain transfer task that the parasite aims to solve.

Parasite Network The parasite network is a much smaller network than the host network and is the primary vehicle for domain shift. The smaller network allows for efficient domain shift as we only need to train a relatively small number of parameters to achieve strong performance. With the name inspired by a biological parasite, the parasite network extracts information at various points from the host network and, in the case of a convolutional neural network setting, concatenates the outputs along the channel dimensions. We experimentally found that concatenating the two outputs performs better than simply adding them together. The key difference between the side-tuning approach and the parasite approach is that side-tuning combines information at the end of the respective networks while the parasite approach extracts information at multiple intermediate points of the original network. The parasite model parameters are trained on the new domain task while implicitly deciding which pieces of information from the host is important.

4 Experiments

Datasets and Implementation Details. To show the effectiveness of the parasite network approach, we characterize its performance against both the standard substitutive approaches and additive approaches described in Section 2. Focusing on domain transfer for CNN foundation models, the parasite method was compared against other methods for CNNs. Scores were taken and replicated from Jia et al. [17] official github repository with their paper reporting umbrella task averages for ResNet-50s. The pretrained model was torchvision's ResNet-50 trained on ImageNet-1K [21, 10]. To set up experiments, we follow the methodology of Jia et al. [17] and their VPT method for CNNs. We use six image classification task datasets from the VTAB-1K datasets with approximately 1000 training examples each: 1) Oxford-IIIT Pet (Pet) [23], 2) 102-Flowers (Flowers) [22], 3) EuroSAT [16], 4) CIFAR-100 (CIFAR100) [19], 5) Dmlab [29], and 6) Describable Textures Dataset (DTD) [9]. Images were reshaped to 224x224 pixels and normalized before passed into the model.

For efficient comparison, we followed the training details as Jia et al. [17]. We used AdamW with a cosine-decay scheduler and 10 warm-up epochs. We followed the learning rate scaling rule as described in Krizhevsky [18] and Goyal et al. [13] and set the true learning rate (μ_{true}) equal to

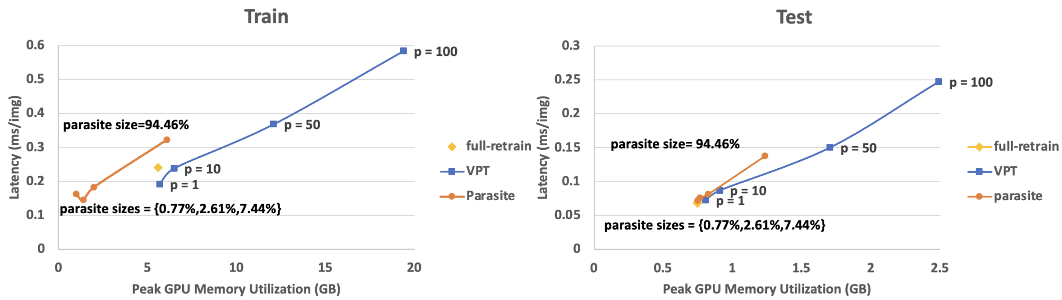

Figure 2: VPT, Parasite, and Full-Retrain peak GPU memory utilization and average latency were compared for both train (left) and test (right). Parasite sizes are the percent size of a ResNet-50.

a function of the base learning rate (μ_{base}) and the batch size (B): $\mu_{true} = \mu_{base} * \frac{B}{256}$. We train for 100 epochs and report the average over three runs. We trained 18 parasites of various sizes and reported results for parasites around 2.61% (617K parameters) and 7.44% (1.8M parameters) the size of a ResNet-50. To create the parasite, the architecture of a ResNet-50 was maintained but the number of parameters were reduced by taking a ratio of the number of in- and out-channels. Additional in-channels were added to the convolutional layers that take in the outputs of both the parasite and the intermediate layers of the host. Instead of random initialization, the weights of the parasite were initialized to a small portion of the original host network as it led to marginal improvements in the final scores. We perform the same grid-search as Jia et al. [17] by varying the batch size, base learning rate (base_lr), and weight decay to determine the best configuration for each parasite size and dataset combination. Configurations can be found in supplementary materials.

Results. Table 2 shows that the parasite model outperforms all other baselines on average. The relative performance between the parasite and the linear probes show that multiple points of information from the frozen backbone can lead to improved performance. The parasite method outperforms all additive methods except VPT and has a much smaller average standard deviation over the datasets than side-tuning, giving greater confidence in our results. The parasite performs as well as VPT on three out of six datasets and outperforms on the remaining three datasets.
To see how the parasite adapts without any real host, we train a parasite of size 2.61% of a ResNet-50 that accepts only zeros from the host (zeroed host). As shown in Table 1, this model greatly underperforms suggesting that the host is needed for successful domain transfer. A disadvantage of the parasite approach is that a weak, too out of domain host model can't extract quality information for domain shift leading to poor performance. However, a strong host can lead to strong transfer of tasks.

5 GPU Memory Utilization and Latency

To show resource efficiency of the parasite approach, we look at the peak GPU utilization and average latency for both train and testing environments. We compared the parasite approach to full-retraining, and VPT with 1 (106K), 10 (131K), 50 (268K), and 100 (492K) prompt tokens (p). We chose parasites of 0.77% (183K), 2.61% (617K), 7.44% (1.8M), and 94.46% (22.3M) the size of a ResNet-50. We used the same experimental setup and datasets as Section 4 and used a Nvidia V100-32GB GPU and a batch size of 64. The peak utilization and average latency are reported in Figure 2 and more detailed numbers are in the supplementary materials.

The parasite outperforms the other two methods for GPU utilization. The parasite has competitive latency for test and outperforms for training. However, the parasite scales much better than VPT for test latency when considering the range of the number of training parameters in Figure 2.

6 Conclusion

We introduced the parasite network, a side-network approach that leverages all block levels with multiple extraction points along the host network. This allows for a significantly reduced number of training parameters as compared to side-tuning. The parasite network performs at least as well as VPT across various VTAB-1K datasets and has improved GPU utilization and competitive latency.

References

[1] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Conditioned and composed image retrieval combining and partially fine-tuning clip-based features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4959–4968, June 2022.

[2] SH Shabbeer Basha, Sravan Kumar Vinakota, Viswanath Pulabaigari, Snehasis Mukherjee, and Shiv Ram Dubey. Autotune: Automatically tuning convolutional neural networks for improved transfer learning. *Neural Networks*, 133:112–122, 2021.

[3] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL `https://aclanthology.org/2022.acl-short.1`.

[4] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11285–11297. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/81f7acabd411274fcf65ce2070ed568a-Paper.pdf`.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, October 2021.

[6] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[7] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR, 13–18 Jul 2020.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Shuyang Gu, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Clip itself is a strong fine-tuner: Achieving 85.7% and 88.0% top-1 accuracy with vit-b and vit-l on imagenet. *arXiv preprint arXiv:2212.06138*, 2022.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[13] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[16] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207, 2018. doi: 10.1109/IGARSS.2018.8519248.

[17] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[18] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[20] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[21] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[22] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[23] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`.

[28] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Convolutions die hard: Open-vocabulary segmentation with single frozen convolutional clip. *arXiv preprint arXiv:2308.02487*, 2023.

[29] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[30] Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: a baseline for network adaptation via additive side networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 698–714. Springer, 2020.

Supplementary Material

A Grid-Search Configurations

Base LR	{0.001, 0.0001, 0.0005, 0.005, 0.05, 0.1, 0.25, 0.5, 1., 2.5, 5., 10., 25., 50.}
Weight Decay	{0.0, 0.0001, 0.001, 0.01}
Batch Size	{1024, 384, 256, 128, 64, 32}

Table 2: Following Jia et al. [17], we chose the best configuration for each dataset and parasite size combination for our experiments.

B Example Use Case of the Parasite Network

In an example to show how the parasite model would be used, let us say we are performing domain shift from ‘video action recognition’ to ‘movie genre classification.’ Specifically, we have a model that has been trained on the Kinetics-700 Dataset [6], and we are using this pretrained model to detect movie genres such as romance or tragedy. If we are in a resource-constrained environment and want to classify concurrently, we can attach the parasite to the model pretrained on Kinetics. The pretrained model can detect actions while the parasite model can detect genres. Keeping in mind the resource constraints, few additional parameters have been added for domain transfer. Also, peak GPU utilization and average latency remain competitive or better than other methods.

C Architecture Details

For this paper, we followed the ResNet-50 architecture. While the host network is a ResNet-50, parasite networks are different. Parasite networks have the same structure as a ResNet-50, but the number of out channels is chosen as a ratio of the original ResNet-50 layer. The input of a block is the concatenation of the previous block’s output and the host intermediate block’s output. Therefore, there is a small bulge in the input channels at the beginning of the next block with the number of in-channels indicated in Figure 3. The ratio for the figure below is for 0.2 which corresponds to 7.44% (1.8M) the size of a ResNet-50.

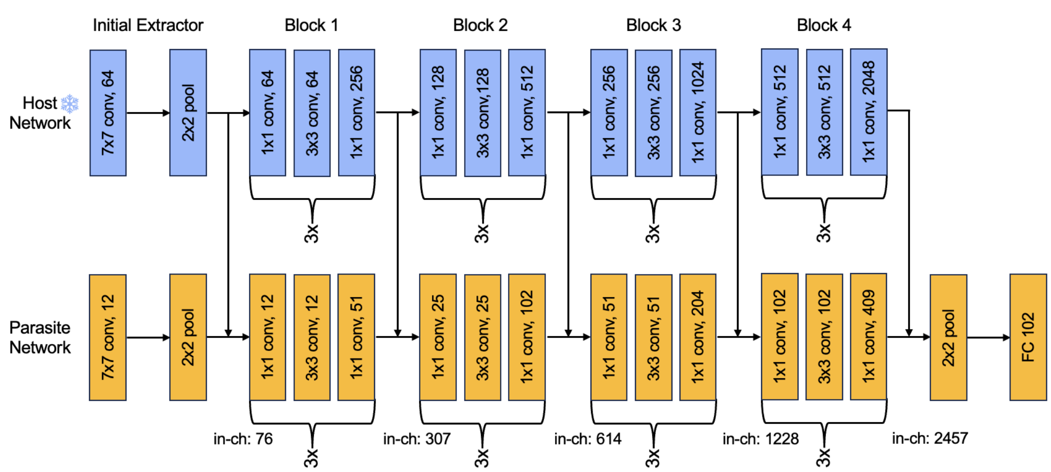

Figure 3: Host Network (ResNet-50) & Parasite Network (ResNet-50 Inspired) Interaction Architecture: Detailed architecture of how the parasite interacts with the host using the Flowers dataset. For this case, the host and parasite network both follow ResNet-50 with the parasite significantly smaller than the host. The number of in-channels at the beginning of each block are indicated with “in-ch” with a small bulge of parameters for the concatenation of the two intermediate block outputs.

D Number of Training Parameters Graph

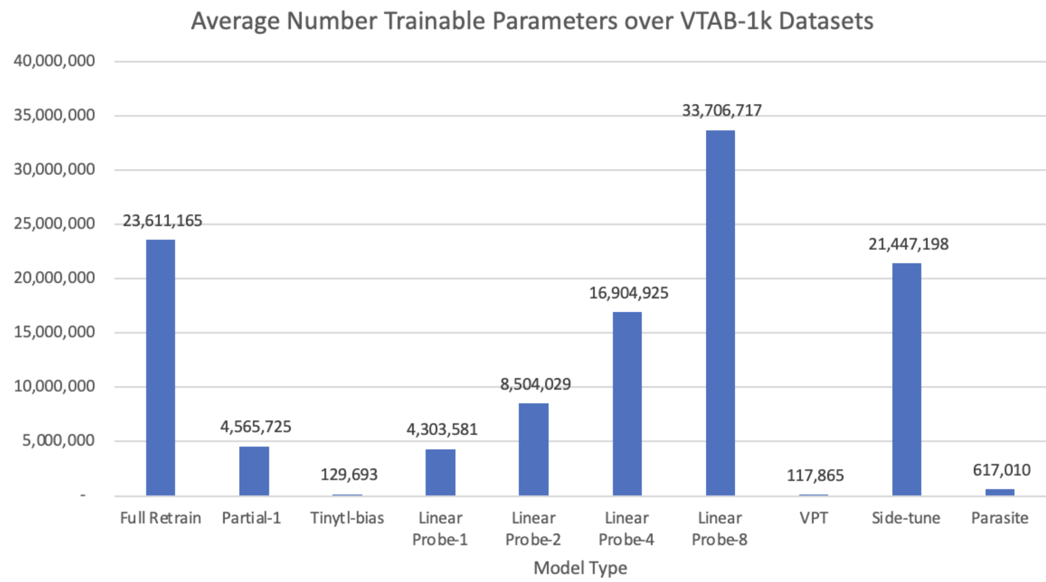

Figure 4: The parasite model requires few additional parameters and is significantly more efficient than side-tuning. The parasite shown in the graph is 2.61% the size of a ResNet-50.

E More Detailed Peak GPU Memory Utilization and Latency Results

		Train Memory (GB)	Train Latency (ms/img)	Test Memory (GB)	Test Latency (ms/img)
Parasite	size = 0.77%	1.0	0.1624	0.7514	0.0717
	size = 2.61%	1.4	0.1449	0.7672	0.0757
	size = 7.44%	2.0	0.1827	0.8255	0.0812
	size = 94.46%	6.1	0.3219	1.2367	0.1379
VPT	p = 1	5.70	0.1911	0.8078	0.0723
	p = 10	6.50	0.2378	0.9093	0.0867
	p = 50	12.10	0.3677	1.7072	0.1501
	p = 100	19.40	0.5833	2.4922	0.2473
Full-Retrain		5.60	0.2403	0.7489	0.0683

Table 3: Comparison of parasite network, VPT, and full-retrain's GPU peak memory utilization (GBs) and avg latency (ms/img) for train and test. Results are reported for a batch size of 64. The parasite size is the percent size of a ResNet-50 and "p" refers to the number of prompt tokens. These results are used to create the graphs in Figure 2.

F Adjusting Number Trainable Parameters

The main results of the paper explore the parasite model at ratio sizes 2.61% (617K) and 7.44% (1.8M) of a ResNet-50. In order to choose the best model, we ran 18 configurations of parasites at various sizes. We looked at how the parasite model performs when decreasing or increasing the number of trainable parameters or size of the parasite network. Following the same implementation details and datasets as Section 4, multiple parasite networks were trained across various parameter space sizes with a ResNet-50 as the host model. The results can be found in Figure 5.

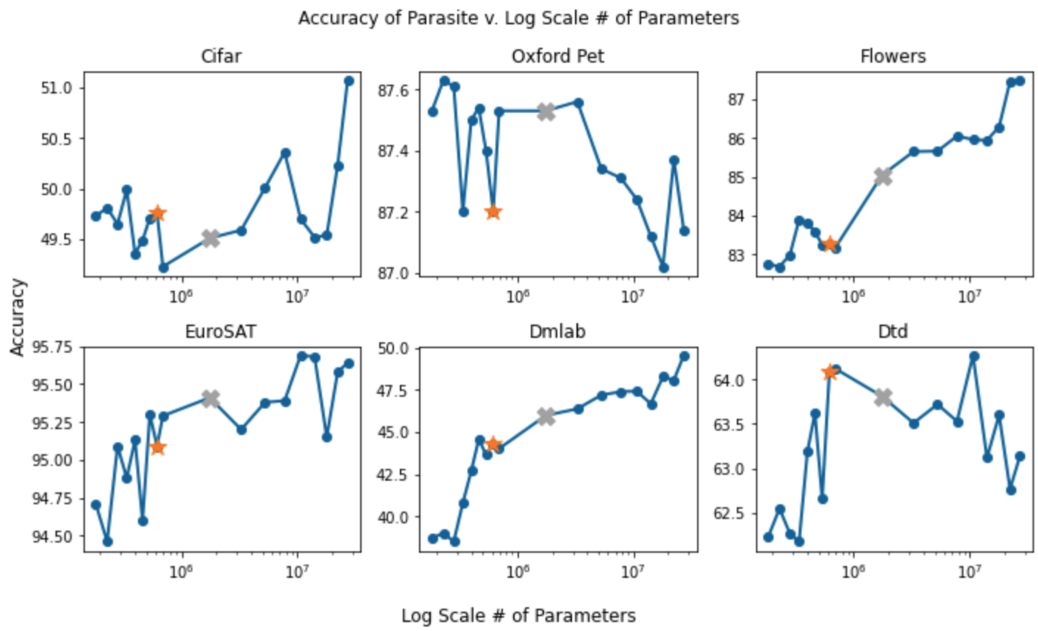

Figure 5: The number of parameters are shown on a log-scale with non-logged amounts bounded between 182.6K and 27.2M additional parameters. The orange star and the gray "X" indicates 2.61% and 7.44% parasite sizes respectively which were reported in the main experiments and in Table 1.

Figure 5 shows that the scores do not rapidly decline (or in some cases improve) even if the number of parameters are driven to as low as 0.77% (182.6K) the size of a ResNet-50. A parasite network was chosen to be 2.61% the size of ResNet-50 (617K) for the experiments in this paper as this setup offered a good performance and efficiency trade-off. Users may require a different performance/efficiency trade-off point, such as 7.44% based on the storage requirement constraints and acceptable accuracy performance of the application. We therefore also report these results in the main experiments.